

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 6 :

G06F 1/00, G08B 25/01

A1

(11) International Publication Number:

WO 96/15485

(43) International Publication Date:

23 May 1996 (23.05.96)

(21) International Application Number: PCT/CA95/00646

(22) International Filing Date: 15 November 1995 (15.11.95)

(30) Priority Data:

08/339,978

15 November 1994 (15.11.94) US

(60) Parent Application or Grant

(63)*Related by Continuation

US

Filed on

08/339,978 (CIP)

15 November 1994 (15.11.94)

(71) Applicant (for all designated States except US): ABSOLUTE SOFTWARE CORPORATION [CA/CA]; Dept. 121, 101-1001 West Broadway, Vancouver, British Columbia V6H 4B1 (CA).

(72) Inventors; and

(75) Inventors/Applicants (for US only): COTICHINI, Christian [CA/CA]; 6-1035 West 12th Avenue, Vancouver, British Columbia V6H 1L5 (CA). CAIN, Fraser [CA/CA]; 5-1035 West 12th Avenue, Vancouver, British Columbia V6H 1L5 (CA). NGUYEN, Thanh, Cam [CA/CA]; 1501 6th Avenue, New Westminster, British Columbia V3M 2C5 (CA).

(74) Agents: ROBINSON, Christopher, J. et al.; Fetherstonhaugh & Co., Suite 1010, 510 Burrard Street, Vancouver, British Columbia V6C 3A8 (CA).

(81) Designated States: AL, AM, AT, AU, BB, BG, BR, BY, CA, CH, CN, CZ, DE, DK, EE, ES, FI, GB, GE, HU, IS, JP, KE, KG, KP, KR, KZ, LK, LR, LT, LU, LV, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, TJ, TM, TT, UA, UG, US, UZ, VN, European patent (AT, BE, CH, DE, DK, ES, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, ML, MR, NE, SN, TD, TG), ARIPO patent (KE, LS, MW, SD, SZ, UG).

Published

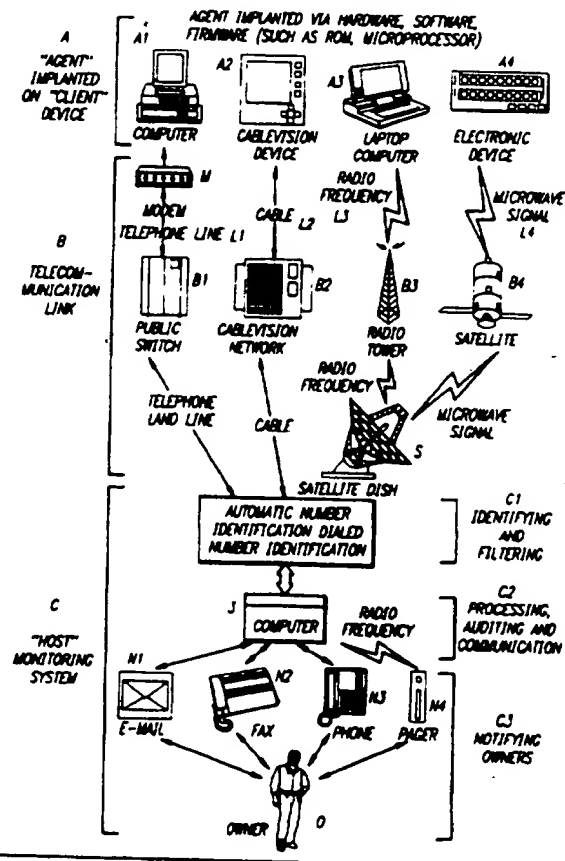
With international search report.

Before the expiration of the time limit for amending the claims and to be republished in the event of the receipt of amendments.

(54) Title: SECURITY APPARATUS AND METHOD

(57) Abstract

A system for locating and monitoring electronic devices utilizing a security system that is secretly and transparently embedded within the software, firmware, or hardware of a computer. This security system causes the client computer to periodically and conditionally call a host system to report its serial number via an encoded series of dialed numbers. A host monitoring system receives calls from various clients and determines which calls to accept and which to reject. This determination is made by comparing the decoded client serial numbers with a predefined and updated list of numbers corresponding to reported stolen computers. Only calls from clients on the predefined list are accepted. The host also concurrently obtains the caller ID of the calling client to determine the physical location of the stolen device. The caller ID, indicating the physical location of the client computer, and the serial number are subsequently transmitted to a notifying station in order to facilitate the recovery of the stolen device. The security system remains hidden from the user, and actively resists attempts to disable it.



FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AT	Austria	GB	United Kingdom	MR	Mauritania
AU	Australia	GE	Georgia	MW	Malawi
BB	Barbados	GN	Guinea	NE	Niger
BE	Belgium	GR	Greece	NL	Netherlands
BF	Burkina Faso	HU	Hungary	NO	Norway
BG	Bulgaria	IE	Ireland	NZ	New Zealand
BJ	Benin	IT	Italy	PL	Poland
BR	Brazil	JP	Japan	PT	Portugal
BY	Belarus	KE	Kenya	RO	Romania
CA	Canada	KG	Kyrgyzstan	RU	Russian Federation
CF	Central African Republic	KP	Democratic People's Republic of Korea	SD	Sudan
CG	Congo	KR	Republic of Korea	SE	Sweden
CH	Switzerland	KZ	Kazakhstan	SI	Slovenia
CI	Côte d'Ivoire	LI	Liechtenstein	SK	Slovakia
CM	Cameroon	LK	Sri Lanka	SN	Senegal
CN	China	LU	Luxembourg	TD	Chad
CS	Czechoslovakia	LV	Latvia	TG	Togo
CZ	Czech Republic	MC	Monaco	TJ	Tajikistan
DE	Germany	MD	Republic of Moldova	TT	Trinidad and Tobago
DK	Denmark	MG	Madagascar	UA	Ukraine
ES	Spain	ML	Mali	US	United States of America
FI	Finland	MN	Mongolia	UZ	Uzbekistan
FR	France			VN	Viet Nam
GA	Gabon				

SECURITY APPARATUS AND METHOD

BACKGROUND OF THE INVENTION

Many electronic devices, such as laptop computers and cellular telephones, are becoming more compact and portable. While such portability is extremely convenient for the user, it has given rise to an increased risk of theft. These electronic devices are often very expensive and are easily lost or stolen.

5 Previously, attempts have been made to provide means for retrieving lost or stolen items of various types. The simplest approach is marking the item with the name and the address of the owner, or some other identification such as a driver's license number. If the item falls into the hands of an honest person, then the owner can be located. However, this approach may not deter a thief who can remove visible markings on the device.

10 Password protection schemes are of dubious value in discouraging theft or retrieving an item. Although the data can be protected from theft, the computer hardware cannot be found or retrieved. Another approach has been to place a radio transmitter on the item. This has been done in the context of automobile anti-theft devices. The police or a commercial organization monitors the applicable radio frequency to try to locate a stolen vehicle. This method is not
15 suitable for smaller items such as cellular telephones or laptop computers. First, it is inconvenient to disassemble such devices in order to attempt to install a transmitter therein. Second, there may not be any convenient space available to affix such a transmitter. Furthermore, a rather elaborate monitoring service, including directional antennas or the like, is required to trace the source of radio transmissions.

20 It is therefore an object of the invention to provide an improved means for tracing or locating smaller lost or stolen objects, particularly laptop computers, cellular telephones, desktop computers and other small, portable electronic devices or expensive home and office electronic equipment. It is also an object of the invention to provide an improved means for tracing such electronic devices which can be installed without disassembly or physical alteration of the devices
25 concerned.

It is a further object of the invention to provide an improved means for locating lost or stolen items, this means being hidden from unauthorized users in order to reduce the risk of such means being disabled by the unauthorized user.

30 It is a still further object of the invention to provide an improved means for locating lost or stolen items which actively resist attempts to disable the means by an unauthorized user.

SUBSTITUTE SHEET

It is a still further object of the invention to provide an improved means for inexpensively and reliably locating lost or stolen items.

5 The invention overcomes disadvantages associated with the prior art by yielding a security device for small computers, cellular telephones or the like which can be programmed onto existing memory devices such as ROM devices, hard disks or the like. Accordingly, no physical alteration is necessary or apparent to a thief. The existence of the security device is well cloaked and it cannot be readily located or disabled even if the possibility of its existence is suspected. Apparatuses and methods according to the invention can be very cost effective, requiring relatively inexpensive modifications to software or hardware and operation of relatively
10 few monitoring devices.

SUMMARY OF THE INVENTION

15 This invention, Electronic Article Surveillance System, relates to a security apparatus and method for retrieving lost or stolen electronic devices, such as portable computers. This invention enables electronic articles to be surveyed or monitored by implanting an intelligent Agent with a pre-defined task set onto an electronic device. This Agent communicates with a preselected Host Monitoring System which is capable of multiple services including; tracing location, identifying the serial number, and electronically notifying the end user/owner of its
20 location. The Agent hides within the software/firmware/hardware of the electronic device, and operates without interfering with the regular operation of the device. The Agent is designed to evade detection and resist possible attempts to disable it by an unauthorized user.

25 According to one aspect of the invention there is provided an electronic device with an integral security system. The security system includes means for sending signals to a remote station at spaced apart intervals of time. The signals including identifying indicia for the device. Preferably, the means for sending signals includes a telecommunications interface connectable to a telecommunications system, and means for dialing a preselected telecommunications number. The remote station includes a telecommunications receiver having said preselected telecommunications number.

30 Where the electronic device is a computer, the means for sending signals includes means for providing signals to the telecommunication interface to dial the preselected telecommunication number and send the identifying indicia. The telecommunication interface may include a modem. The means for providing signals may include security software programmed on the computer.

SUBSTITUTE SHEET

The Agent security system may be recorded on the boot sector of a hard disk or, alternatively, on a hidden system file such as IO.SYS, MSDOS.SYS, IBMBIO.COM or IBMDOS.COM.

5 There is provided according to another aspect of the invention a method for tracing lost or stolen electronic devices whereby a telecommunications interface is connectable to a telecommunications system at a first telecommunications station. The method includes providing the electronic device with means for sending signals to the telecommunications interface. The means is instructed by the program to send first signals to the telecommunications interface which dials a remote telecommunications station. These first signals contain the encoded
10 identification (serial number) of the sending computer. The telecommunications interface then dials a remote telecommunications station corresponding to the intended receiving computer. Upon detecting a ring signal, the remote computer retrieves the caller phone number and the identification of the sending computer from the telephone company. The remote computer decodes the serial number of the sending computer, and compares it with a predefined listing of
15 serial numbers of lost or stolen computers. The call will only be answered if the sending computer is on the predefined list.

In an alternative embodiment, if the remote computer answers the ring then the means for sending signals automatically sends second signals to the telecommunications interface, which transmits to the remote telecommunications station identifying indicia for the device as well as
20 any other pertinent information.

There is provided according to another aspect of the invention a method for encoding the serial number of the sending computer within a sequential series of dialed numbers. In this method, a predetermined digit within the dialed number sequence corresponds to one of the digits of the serial number. The preceding digit within the encoded signal indicates which digit within
25 the serial number sequence that the predetermined digit represents.

BRIEF DESCRIPTION OF THE DRAWINGS

30 These and other objects and advantages will become apparent by reference to the following detailed description and accompanying drawings, in which:

FIG. 1 is a functional block diagram of the Electronic Article Surveillance System in accordance with the teachings of this invention.

SUBSTITUTE SHEET

FIG. 2 is a simplified illustration of FIG. 1 for the purpose of showing an illustrative embodiment of the present invention.

5 FIG. 2A is a flowchart of the process by which the operating system and Agent are able to start up and run simultaneously.

FIG. 2B is a flowchart of the process by which the Host Identification and Filtering Subsystem identifies and filters out unwanted calls from Agents.

10 FIG. 2C is a flowchart of the process by which the Host Processing, Auditing and Communication Subsystem, contained within the host computer, exchanges data with an Agent.

15 FIG. 2D is a flowchart of the process by which the Host Notification Subsystem, contained within the host computer, notifies end-users of the status of monitored devices.

20 FIG. 3 is a flowchart showing the conventional method of booting up a personal computer with alternative loading points for the Agent security system shown in broken lines.

FIG. 3A is a flowchart showing a method for startup loading of an Agent security system according to an embodiment of the invention wherein the operating system boot sector is loaded with the Agent.

25 FIG. 3B is a flowchart similar to FIG. 3A wherein the hidden system file IO.SYS or IBMBIO.COM is modified to be loaded with the Agent.

FIG. 3C is a flowchart similar to FIG. 3A and 3B wherein the partition boot sector is modified to be loaded with the Agent.

30 FIG. 3D is a flowchart similar to FIG. 3B and 3C wherein the Agent security system is ROM BIOS based.

SUBSTITUTE SHEET

FIG. 3F, 3G are portions of a flowchart showing the Agents' work cycle apparatus and method according to an embodiment of the invention.

FIG. 3H is an isometric view, partly diagrammatic, of the physical structure of a computer disc.

FIG. 4 is a schematic showing the encoding/decoding method whereby the monitoring service would have to subscribe to 60 telephone numbers.

FIG. 4A is a schematic showing the encoding/decoding method whereby the monitoring service would have to subscribe to 300 telephone numbers.

DESCRIPTION OF THE PREFERRED EMBODIMENTS

System Overview

Referring to Figure 1, the Electronic Article Surveillance System is comprised of three main components: (1) Client device A consisting of any electronic device which has been implanted with the Agent; (2) A telecommunication link B such as a switched communications system, cable networks, radio/microwave signal; and (3) The host monitoring system C which controls the communications between the client device A and the host monitoring system C.

Referring to FIG. 1, the client device can be a cablevision device A2, laptop computer A3, or other type of electronic device A4. However, for illustrative purposes, the client device consists of a computer A1 attached to modem M. The host monitoring system C sends and receives data packets from the client computer 10 over a suitable bi-directional transmission medium, such as a common telephone line L1. Telephone line L1 couples the client device C to the host monitoring system C, and the host computer 3, through Public Switch B1 (telephone company). The host monitoring system C notifies the appropriate parties C3 (owner O, law enforcement agency, or monitoring company) of the status of the client device A via suitable communication means such as electronic mail N1, fax N2, telephone N3 or pager N4. Host monitoring system C also identifies and filters incoming calls C1, and also provides processing, auditing and communication functions C2.

SUBSTITUTE SHEET

In another embodiment of the invention cablevision device A2 is connected to cablevision network B2 via cable L2. This cable L2 further connects cablevision network L2 to the host monitoring system C.

5 In another embodiment of the invention laptop computer A3 is connected to radio tower B3 via radio frequency (RF) transmissions L3. These RF transmissions are received by satellite dish S at the host monitoring system C.

In yet another embodiment of the invention electronic device A4 is connected to satellite B4 via microwave signal L4. Microwave signal L4 further connects satellite B4 to satellite dish S at the host monitoring system C.

10 Referring to FIG. 2, the Host Monitoring system C is comprised of a Voice Board 2, Host Monitoring Computer 3, Hard Disk Controller 4, Hard Disk 5, CRT 6, Keyboard 7, and Printer 8. The host monitoring computer 3 is coupled to a suitable display device, such as a CRT monitor 6, keyboard 7, and to printer 8. The keyboard 7 permits the operator to interact with the Host Monitoring System C. For example, the operator may use keyboard 7 to enter
15 commands to print out a log file of the clients that have called into the system. The host computer 3 illustratively takes the form of an IBM personal computer. The source codes for the host monitoring system C, in Visual C++ by MicroSoft, are attached in the Appendix.

Telephone line 1 is connected to the computer 3 by a voice board 2 adapted to receive and recognize the audible tones of both caller ID and dialed numbers transmitted via the
20 telephone line 1. Client computer 10 is connected to modem 9 via serial ports 9a. Host computer 3 is connected to voice board 2 via serial port 2a. The modem 9 and voice board 2 are connected to telephone line 1 which is routed through public switch 9b in accordance with a conventional telephone system. Computer 10 and modem 9 form a first telecommunication station, while computer 3 and voice board 2 form a second, or remote telecommunications
25 system. The Host Monitoring System C sends and receives data packets from client computer 10.

Ring signals are received on phone line 1 as an input to voice board 2. In an illustrative embodiment of the invention, voice board 2 may take the form of the DID/120, DTI/211 and D/12X Voice boards manufactured by Dialogic Corporation. The voice board 2 is coupled to
30 host computer 3 via data bus 2a. The voice board 2 is operative to recognize the ring signal. Then it receives the caller ID and dialed numbers and converts them into corresponding digital signals. Host computer 3 uses these signals for comparison against a list stored in hard disk 5.

SUBSTITUTE SHEET

5 In an illustrative embodiment of the invention, the hard disk controller 4 may comprise memory control boards manufactured by Seagate Tech under the designation Hard Disk Controller. The hard disk controller 4 is particularly suitable to control the illustrative embodiment of the hard disk memory 5 manufactured by Seagate Tech under their designation ST-251.

10 The Agent is a terminated and stay resident program which is installed on hardware, software, or firmware. The alternative methods of installation are described in detail in FIGS. 3A, 3B, 3C, and 3D. Once the Agent is installed it will report its identity and its location to the host after specified periods of time have elapsed, and upon the occurrence of certain predetermined conditions. This is further illustrated in FIG. 2A. Client source codes are disclosed, in Tazam Assembler Code by Borland, in the Appendix.

Installing and Loading the Agent

15 The Agent is installed during a typical boot up sequence to the operating system of a computer. FIG. 3 shows a boot-up process for a typical personal computer. The details of the boot up process are discussed in Appendix I. It should be understood that this invention is applicable to other types of computers and electronic devices presently available or as marketed in the future with suitable modifications. The aspect of the invention described below is the process of installing the security software onto a portable computer such as client computer 10.

20 The method of installation is crucial because the software must remain undetectable once installed. Furthermore, the software should be as difficult as possible to erase. In summary, the invention achieves these objects by installing the software in such a manner that it remains hidden to the operating system, such as MS-DOS.

25 Three alternative ways of installing the Agent security system during the disk boot are illustrated in FIG. 3A-3C respectively. A conventional boot up method is described in detail in Appendix I. A fourth alternative, installing via ROM, is shown in FIG. 3D. The system can also be installed with MS.SYS or IBMDOS.COM, but these are more difficult and less preferred than the three alternatives set out below. The loading program TENDER (further described in the Appendix) can be used to install the Agent by one or more of these alternative installation

30 methods. Thus, the Agent may be installed in a variety of locations whereby second and third Agents can provide back up support for the primary Agent. The three locations where the Agent can be installed on the client device are as follows:

1. The operating system boot sector- See FIG. 3A.

SUBSTITUTE SHEET

2. A hidden system file such as IO.SYS for MS-DOS or IBMBIO.COM for PC-DOS- See FIG. 3B.
3. The partition boot sector- See FIG. 3C.

5 Referring to FIG. 3A, the Agent loading sequence is described for loading the Agent on the operating system boot sector. The computer 10 is powered on and the loading sequence begins 64. As is well known in the art, the computer 10 performs an initial testing routine to assure that all components are working properly 65. Illustratively, the program incorporated is the IBM-PC compatible Power-On Self Test (POST) routine. The partition boot sector is loaded 10 66. Next the operating system boot sector with the installed Agent is loaded 67. In an effort to maintain the transparency of the Agent, the CPU registers (corresponding to the current state of the computer) are saved 68. Before the Agent is installed there is a check for a Remote Procedure Load (RPL) signature 69. If the signature is present this indicates that the Agent is already in memory and will not be loaded again. However, if there is no RPL signature then 15 preparation is made to load the Agent. First, space is reserved for the Agent at the ceiling of conventional memory 70. Next, Interprocess Communication Interrupt (2Fh) is hooked 71 which enables communication with other programs. Interrupt 13h, which is the disc input/output handler, is hooked 72. The old timer interrupt is saved, and new hook timer interrupt is put into place 73. Now the CPU registers are restored 74 in order to maintain the transparency of the 20 system. The original operating system boot sector is loaded 75. The original operating system had been moved to accommodate the Agent installation. Finally, the operating system is loaded 76 and running 77 again.

Referring to FIG. 3B, the Agent loading sequence is described 78-91 for loading the Agent on a hidden system file such as IO.SYS for MS-DOS or IBMBIO.COM for PC-DOS. 25 The sequence is analogous to that disclosed above for the operating system boot sector. However, instead of lading the Agent with the operating system boot sector, the Agent is loaded with the operating system file 82 (load modified IO.SYS or IBMBIO.COM).

Referring to FIG. 3C, the Agent loading sequence is described 92-104 for loading the Agent on the partition boot sector. The sequence is analogous to that disclosed above for the 30 operating system boot sector. However, instead of loading the Agent with the operating system boot sector, the Agent is loaded with the operating system partition boot sector 94.

Referring to FIG. 3D, the Agent loading sequence is described 105-116 for loading the Agent via ROM BIOS. This schematic illustrates an embodiment of this invention on firmware. The sequence is analogous to that disclosed above for the operating boot sector. However, the

SUBSTITUTE SHEET

Agent is loaded from the ROM after the CPU registers are saved 107. At that time the ROM can take control of the system and load the Agent. Once the CPU registers are restored 113, the ROM can no longer load the Agent.

FIG. 2A is a flow chart of the Agent Work Cycle. This Work Cycle describes the method by which the Agent is loaded when the computer 10 is initially turned on, and the manner in which the operating system and the Agent run simultaneously. Once the client computer 10 is powered on 11, it performs a power on self-test (POST) 12. The POST tests the system hardware, initializes some of the devices for operation, and loads the master boot record (MBR) 13. Since the MBR was installed with an Agent Subloader, the Subloader is loaded into memory 14 and executed. The Subloader's first task is to load the Agent 15 into memory. Then the Subloader loads the operating system (OS) into memory 16 and returns control to the operating system. Now both the operating system 17 and the Agent 18 are running simultaneously.

Functions of the Agent

Referring to Figure 2A, the Agent's primary job is to determine the appropriate time for it to call the Host Monitoring System (Host) 19 to report its status (such as identity, location and other information). Secondly, like any terminated and stay resident program, the Agent will not interfere with any running applications unless designed to interfere. Thus, the Agent can avoid being detected. The Agent will determine if it should call the Host 18 times per second. The Agent will only call the host when a pre-defined time period has elapsed, or a pre-determined event has occurred which triggers the client to contact the host. The Agent compares the current date and time with the date and time corresponding to the next time that the Agent is due to call the host. If the Agent determines that it is time to call the Host, it will do a thorough search within the computer 10 to find free (not currently being used by any running application) communication equipment 20. In an illustrative embodiment, the communication equipment is a modem 9. If the agent fails to find any free equipment, then it will abort its attempt to call the Host and repeat the cycle 18. However if the Agent locates free communication equipment, it will call the Host 21. Upon receiving a call from the client 10, the Host examines the Agent identity and determines if a connection should be established 22. If the Host does not accept the call then the Agent will not call back until the next appropriate time (after predetermined time period has elapsed) 18. If the Host accepts the call, then the Agent will send the Host its encoded identity (serial number), location (caller ID) and any other pertinent information such as local date and time 23. The Agent then checks if the Host has any

SUBSTITUTE SHEET

data or commands for the client 24. If the Host has no data or commands to be sent, then the Agent will terminate the call and repeat the cycle 18. Otherwise, the client will receive the data or commands from the Host before it terminates the call and repeats the cycle 18. This Work Cycle is described in much greater detail in FIGS. 3F and 3G and is described in the Detailed Operation section.

The system remains transparent to an unauthorized user via implementation of well known deflection methods. Attempts to read or write to the location where the Agent has been installed are deflected in order to prevent discovery of the Agent. When read attempts are made to the Agent location the system generates meaningless bytes of data to be returned to the user. When write attempts are made to the location where the Agent is installed, the client computer accepts the input data and informs the user that the write has been successful. However, the data is not really stored, and thus the Agent is preserved. In the Appendix, the source code for the disk deflection routines are disclosed within file SNTLI13V.ASM.

Detailed Operation of Agent Work Cycle

Referring to FIG. 3F, the following is a description of what happens during the period of time when the Agent security system is in "active" mode 117, 118:

Once the system is powered on, the timer interrupt will occur 18.2 times per second 117. Every 18 timer interrupts, the complementary metal-oxide semiconductor (CMOS) real-time clock will be accessed, and the time and date will be stored for comparison with the previous real-time clock access. If the date and/or time changes towards the future, no action will be taken to track the time displacement. In this way the Agent determines whether it is time to call the host 118. Thus if the current date has advanced far enough into the future (past the date and time to call the host), the Agent security system will change its mode of operation from active to alert 119 whereby calls will be regularly attempted until a call is made and a transaction with the host server has been completed. If the system time has been backdated, this will also cause a modal change from active to alert.

Referring to FIGS. 3F and 3G, the following is a description of what happens when the Agent security system is in "alert" mode 119-161:

The communications ports are checked 119-125 (via a port address table 120) to see if they exist. If the first one encountered is not in use 123, it will be dynamically hooked 126 into by swapping the appropriate interrupt handler and unmasking the appropriate interrupt request line. If an error occurs, the next port will be checked 124 until either a valid port is found or

SUBSTITUTE SHEET

the port address table has been exhausted 125. Appropriate cleanup routines restore "swapped" ports to their initial settings.

5 If the communications port responds properly, the system will then attempt to connect to a modem via issue of the Hayes compatible AT command 128. If the modem does not exist, then the next port will be checked 124. If the modem responds with an 'OK' to the AT command 129, the system will attempt to initialize the modem by sending it a modem initialization string 130, 132 (from a table of initialization strings 131). If the modem does not respond with an "OK" 134, this indicates that the initialization attempt failed 135. If the initialization attempt failed, then the next string in the table will be tried 136, and so on until a valid initialization string is found 134, or the modem initialization string table is exhausted 136 (at which point, the routine will delay for some seconds then try again from the start, using the first initialization string 130).

10 Once a valid and available communications port has been found, and it has been verified that a functional modem is associated with that port, the system will attempt to dial out to the remote host server 137, 138.

15 A dial string table 140 is used 139 to attempt the call since a PBX or switchboard etc. may need to be exited via a dialing prefix. If successful 141-143, the CONNECT result code (numeric or letters) from the remote host server will be received by the client 143. The host will send a signal ("Query") to the client requesting its serial number. If the client does not receive the query signal 148 it will abort 149 and repeat the cycle 119. If the client receives the "Query" signal, then the serial number is sent 151. At this point, telecommunications have been established and the client-server transaction begins. If the transaction succeeds, the resultant state will be "active", otherwise "alert". If, for some reason, a "NO DIALTONE" event happens 144, a delay will occur 147 and the next dial string 141 will be attempted. If the line is "BUSY" 145, then a redial attempt 146 will occur using the same dial string for a predefined number of attempts or a telecommunications connection is made, whichever comes first.

20 The client to remote host server transaction involves the sending of the computer serial number 151 via the telephone company or carrier service. The "Caller ID" is implicitly received by the remote server (typically during the initial telecommunications event known as "RING"). 25 Upon the telecommunications event called "CONNECT", the remote host server sends the Agent security system client a vendor specific message called "QUERY" 148 which in effect tells the client to send the serial number. The sending of this serial number 151 involves the server acknowledging that it has indeed received 152 and processed 154 the serial number (validating it). The client computer will attempt to send this serial number a predefined number of times 153

SUBSTITUTE SHEET

before it gives up (disconnect, cleanup, unhooks port 127, 155 and returns to "alert" mode 156). At this point, the modem disconnects 160. Any other cleanup necessary (such as changing the date of the last call to the present) will also be done here 160. Finally, the resultant state will be reset to active 161.

5 If the computer that called in was not reported stolen, no further action with regard to the computer system that called in will be taken. If, however, the serial number transmitted to the remote host server matches one of the serial numbers on a currently valid list of stolen computers, further processing will occur to facilitate the recovery of the missing equipment. Such processing includes, but is not limited to, placing either an automatic or manual call to the local
10 authorities in the vicinity of the missing equipment or the owner of such equipment.

Host Identification and Filtering System

15 The Host Identification and Filtering System identifies and filters out unwanted calls from Agents. FIG. 2B is a flow diagram of the Host Identification and Filtering program executed by host computer 3. Once the security program is executed 26, the voice board waits 27 for the ring signal on the telephone line 1. When a ring signal is detected 28, the voice board 2 acknowledges the incoming call by sending a signal to the telephone company 9B via telephone line 1 requesting that the caller ID and the dialed numbers be sent to it. The voice board then waits until these numbers are received 29, 30.

20 Once the caller ID and the dialed numbers have been received, they are saved to the hard disk 31, 32. The security program then compares the dialed numbers 33, which provide a coded version of the serial number of the client computer 10 (coding scheme explained in detail below), against a list of serial numbers stored on the hard disk 4. If no match is found, the program lets the phone ring until the client computer 10 hangs up the telephone line 1. In the preferred
25 embodiment, the client computer is programmed to hang up after 30 seconds of unanswered ringing. However, if a match is found, the security program routes the call to an appropriate receiving line connected to a modem 35, which answers the call.

Encoding of the client computer serial number

30 Referring to FIG. 4, the serial number of client computer 10 is encoded within the dialed numbers it sends to the host 3. In the preferred embodiment of the invention, the client computer transmits its six digit serial number 170 to the host via a series of six complete dialed phone numbers 172. The first eight dialed digits after the first "1" are meaningless. The ninth dialed digit "N" 175, indicates which digit position within the serial number that the tenth dialed

SUBSTITUTE SHEET

number corresponds to. The tenth dialed digit "D" provides the Nth digit of the serial number. The host computer 3 receives the six complete dialed phone numbers 172 and decodes them 173 by looking at only the ninth and tenth dialed digits. The client computer serial number 174 is thus reproduced.

5 For example, in the sequence "800-996-5511", the only relevant digits are the "11" portion. The first "1" indicates that the digit immediate to its right (1) is the first digit in the serial number. Similarly, in the sequence "800-996-5526", the "2" indicates that the number immediate to its right (6) is the second number in the serial number. The client 10, in total, dials six numbers 172 in order to convey its six-digit serial number to the host.

10 In order to accommodate this method of serial number coding, the host monitoring system needs to subscribe to sixty different phone numbers. All sixty numbers should have the same first eight digits, and only vary from one another with respect to the last two digits. The ninth digit need only vary from "1" through "6" corresponding to the six digits within a serial code. However, the last digit must vary from "0" to "9".

15 Referring to FIG. 4A, the coding system can alternatively be modified such that the client computer 10 need only call the host three times to convey its serial number 180. According to this coding method, two digits of the serial number 186 would be transmitted in each call. Thus, the eighth dialed digit 185 would vary from "1" to "3", corresponding to the three packets of two digits 186 that make up the serial number 180. The ninth and tenth dialed digits 186 would vary from "0" through "9". However, this would require the operator of the
20 monitoring system to subscribe to three hundred different phone numbers.

Host Processing, Auditing and Communication Subsystem

25 Referring to FIG. 2C, the Host Processing, Auditing and Communication Subsystem receives and transmits information to and from clients. FIG. 2C is a flow diagram of the Host Communication program executed by host computer 3. After the host computer 3 is powered on 36, communication equipment is instructed to wait 37 for the telecommunication begin signal from the client computer 10. The telecommunication equipment acknowledges the begin signal by initiating a session to communicate with the client computer 38. The program first checks
30 the client computer 39 to establish that it is sending data packets 40, and then receives the packets 41. Next, the program determines if the client has any data or commands to be sent to the host 42. If not, the session is terminated 43, and the cycle is repeated 37. When all data packets have been received, the program permits the host to send data packets to the client computer. The program prepares to send data packets 44, and then establishes that there are

SUBSTITUTE SHEET

more data packets to be sent 45 before sending each packet 46. Once all data packets have been sent, the program terminates the session 43, hangs up the phone, and prepares to repeat the entire cycle 37. Host-side source codes are disclosed in the Appendix in Visual C++ (Microsoft) Code.

5

Host Notification Subsystem

The Host Notification Subsystem notifies the end-users regarding the status of their electronic devices. In FIG. 1, various methods of notification such as; electronic mail N1, fax N2, paging N4, and telephone call N3, are depicted. FIG. 2D is a flow diagram of the Host Notification program executed by host computer 3. The Host Notification program determines whether there are any pending notification instructions or commands 48. If there are pending notifications, the information is retrieved 49. The program then determines the preferred preselected notification method 50, and formulates the message to be dispatched 51 according to the preselected notification method. This message is dispatched to the end-user 52. After dispatching the message, the program repeats the entire cycle 47. Host-side source codes are disclosed in the Appendix in Visual C++ (Microsoft) Code.

10

15

Variations and Alternatives

The above description relates to the Agent security system installed and operating in a conventional PC with an Intel 80X86 microprocessor or equivalent and with a conventional MS-DOS or PC-DOS operating system. It will be recognized that the system can be modified to fit other types of computers including, for example, those sold under the trademark Macintosh. The system can easily be modified to suit other types of operating systems or computers as they develop in this rapidly advancing art.

20

25

The above system is also intended to be added to existing computers without physical alteration. Another approach is to modify the ROM of such computers to contain the Agent security system as shown in FIG. 3D. This is generally not considered to be feasible for computers sold without the security feature, but is a theoretical possibility. More likely is the possibility of incorporating the Agent security system into the ROM of portable computers, cellular telephones or other such items when they are manufactured. FIG. 3D above describes the loading of the system from such a modified ROM.

30

SUBSTITUTE SHEET

The description above also assumes that the computer device has a modem connected thereto or includes an internal modem. In the future it is likely that telephone systems will be digitized, thus obviating the need for a modem.

5 The system could also be included in the ROM of a cellular telephone. In this case, the program should be designed to hide the outgoing calls from the user by silencing audio signals and maintaining a normal screen display. It is also conceivable that portable computers can be supplied with integral cellular telephones modified in this manner or with some other telecommunication device. It is not clear at the time of this invention exactly which direction the field of telecommunications will likely go in the immediate future. The main telecommunication
10 criteria for this Agent security system is that the outgoing transmission (wire, radio signal or otherwise), be received by a switching mechanism, and contain information that causes the switching mechanism to forward the information received to a remote station. Presently, this information is a telephone number. But other indicia of the remote station may be substituted in alternative switchable communications systems.

15 Attached hereto are appendices relating to the following: (1) Description of the conventional boot up method; (2) Details of agent installation; (3) Brief description of the routines; and (4) Copy of the source code of both the client-side and host-side. The host-side source code is in Visual C++ (MicroSoft). The client-side source code is in Tazam Assembler
20 Code by Borland.

It will be understood by someone skilled in the art that many of the details described above are by way of example only and are not intended to limit the scope of the invention which is to be interpreted with reference to the following claims.

SUBSTITUTE SHEET

APPENDIX I - CONVENTIONAL BOOT UP METHOD

Referring to FIG. 3H, an isometric view of a computer disc is shown. This figure illustrates the location of the start of user data 162, partition gap 163, boot sector 164, partition sector 165, and partition gap 166.

Referring to FIG. 3, upon hitting the on switch of a personal computer (PC) 53, the computer first goes through a conventional power-on self-test (POST) 54. At this point the Agent could be loaded if ROM-BIOS loading is used 60. POST ensures that all hardware components are running and that the central processing unit (CPU) and memory are functioning properly. Upon completion of the POST, the next task is to load software onto the random access memory (RAM) of the computer. Conventionally, there is a read-only memory (ROM) device which contains a boot program. The boot program searches specific locations on the hard disk, diskette or floppy disk for files which make up the operating system. A typical disk is shown in FIG. 3H. Once these files are found, the boot program on the ROM reads the data stored on the applicable portions of the disk and copies that data to specific locations in RAM. The first portion of the disk boot sector to be loaded is the partition boot sector 55 shown in FIG. 3H as 165. At this point the load partition boot sector method could be used 61. The partition boot sector 165 then loads the remaining boot sector 164 from the disk, namely the operating system boot sector 56. Now the Agent could be loaded according to the load operating system boot sector method 62. The operating system boot sector 164 loads into memory a system file, normally named IO.SYS on personal computers or IBMBIO.COM on IBM computers 57. Now the Agent could be loaded according to the IO.SYS or IBMMIO.COM methods. Each of these files is marked with a special file attribute that hides it from the DOS Dir. The IO.SYS or equivalent then loads the rest of the operating system, conventionally called MSDOS.SYS on MS-DOS systems, and IBMDOS.COM for PC-DOS systems. Next the AUTOEXEC.BAT is processed and run 58. Now the operating system is running 59. The Agent security system according to the invention is loaded during the boot up process and accordingly is transparent to the operating system.

APPENDIX II - DETAILS OF AGENT INSTALLATION

Once the TENDER program, which enables the Agent to be installed, has been run and the Agent has been determined to be loaded via one, two or three of these alternatives, the system is primed and proceeds to attempt to install the Agent security system according to the present

SUBSTITUTE SHEET

state of the computer memory and the instructions given by the programmer. The SNTLINIT routine initializes the Agent security system and is passed one of three possible loading options via the AX microprocessor register by the calling program (SUBLOADR), which itself was loaded on any one of the three enumerated locations described above. The SUBLOADR program reads the configuration file (which may be encrypted) that was generated for user input. The validity of the configuration file is checked at this point to see if it is corrupted or not. If for some reason it cannot read the configuration file, it initializes the Agent security system from a table of default settings.

The SUBLOADR program then checks to see if the Agent security system is in memory by looking for the "RPL" signature. SUBLOADR saves the application programmer interface (API) entry point and then determines which version of the security program, if any, is in memory. If not in memory, the SUBLOADR program searches the disk for the SNTLINIT routine. Depending upon the version of the SUBLOADR program, it may perform a validity check on the SNTLINIT routine. This routine may be a cyclical redundancy check (CRC) of 16 or 32 bits, a checksum check or a hash count.

The TENDER program checks the partition boot sector, the operating system boot sector, and the IO.SYS (or IBMBIO.COM on PC-DOS systems) to see if any of them have been modified to contain the SNTLINIT code. A comparison to the configuration file is made to determine if the Agent has already been installed in any of the alternative locations. If the Agent has already been installed, the TENDER program takes no action. It then tracks the level of modification that was requested by the user (i.e. whether one, two or three areas were to be modified). Each of these areas has all the modem related information written to it amongst other user selected settings. At this point it writes the current configuration file to disk.

The TENDER program then takes a system snapshot of the partition boot sector, the operating system boot sector and the IO.SYS or IBMBIO.COM file, validating them, determines and then writes this file to disk. It then checks the partition gap between the partitions, calculating the number of unused sectors between the valid boot sectors (be they partition or operating system boot sectors).

There is almost certainly at least 8K of space in the partition gap 163. The Agent security system requires only 4K. The SNTLINIT module is usually stored here. If for some reason

SUBSTITUTE SHEET

there is not enough space in the partition gap, or if the data area is physically unusable, the TENDER program will pick a suitable cluster of sectors, mark the data area logically as being unusable, then store SNTLINIT in the cluster of sectors. The TENDER program sets out the attributes to system, hidden etc in order to hide the program image. It then calculates the physical coordinates of the cluster that was used and writes this information to the configuration file. At this point the system is ready to proceed and will be loaded prior to the completion of the loading of the operating system regardless of what strategy the programmer has chosen.

In a manner similar to how viruses reinfect the boot sector 164 of the hard disk drive, the Agent security system according to the invention uses such technology to help protect against theft of the computer. Other technologies such as system timer programming and communications programming are bound to this virus like technology to create a new technology. It should also be understood that a security company which handles incoming calls from clients may readily redefine the time period between successive calls from a client to its host.

The system is typically in one of two modes of operation: (1) Waiting until it is time to call/report into the server - "active mode"; (2) Calling or attempting to call the server - "alert mode". When the Agent security system changes its mode of operation from active to alert mode, the activation period is reduced to a minimal period such that the Agent calls the host eighteen times per second until a successful connection is made. The activation period in active mode is predetermined, and likely to be days if not weeks. This shortened activation period (time between successive calls) is necessary to prevent busy signals and other temporal error conditions from precluding transaction attempts. The system will stay in this alert mode until a valid transaction has been completed.

Since MS-DOS and PC-DOS were designed to be single-user, single-tasking operating systems, the timer interrupt is used to run the system unattended and automatically in the background to provide multi-tasking. Neither the user nor a potential thief would notice this background process although registered owners will be aware of its existence.

In a standard personal computer, routine housekeeping tasks are performed periodically and automatically by the CPU without instructions from the user. There is a timer routine which is called 18.2 times per second to perform such tasks as turning off the floppy disk motor after a certain period of inactivity. The Agent security system hooks into this timer routine. The total

SUBSTITUTE SHEET

timer routine takes about 55 milliseconds and the Agent security system utilizes a small portion of CPU time during that period; this is limited to less than 0.5% of the total timer routine. This is not sufficient time to run the entire security program. Accordingly, the security program is run in small increments with each timer routine. It is important that the security program not "steal" enough computer time to be noticed. Otherwise the computer would be noticeably slowed and the existence of the program might be suspected.

Serial port and modem setup routines must be called by the timer interrupt. Once this is done, the serial interrupt handler that is being used will handle the details of data transfer between the client and host systems. Once the system is set up, the serial port interrupt handler does most of the work with the timer interrupt acting as a monitor watching the transaction when it happens between the client and the server. It analyzes the receive buffer and takes the appropriate actions as necessary. The communication portion of the system can handle outgoing and incoming data transfers on its own since it has its own access to the CPU via its own interrupt request (IRQ) line, typically IRQ3 or IRQ4. Therefore the system can handle the data flow between the client machine and the server unattended.

At the start of its time-slice, the timer interrupt checks the flag, which is set when a user uses the modem, in the Agent security system, the InComISR flag byte (In Communications Interrupt Service Routine). If the flag is set, the timer interrupt exits immediately so as not to interfere with the progress of any serial communications that may be occurring, therefore not disrupting any transaction in progress. If the flag is not set, the timer interrupt routine will check to see if the Agent security system is in an error state. If not in error, a flag called TimerISR count is set to indicate that a timer interrupt is in progress.

A deferred execution function pointer is used to point to the upcoming routine to be executed. Just before the timer interrupt routine finishes, it points to the next routine to be executed. When the next timer interrupt occurs the routine that was pointed to will be executed. The routine must complete in less than 55 milliseconds so that the next timer interrupt does not occur while the routine is still executing.

Attached to the PC's system bus are communications ports, all of which are optional and typically called COM1, COM2, COM3, COM4 for the first four ports. It is unusual to have more than four serial ports in a PC that is using only MS-DOS or PC-DOS as its operating

SUBSTITUTE SHEET

system. The Agent security system also requires that a modem be connected to one of these serial ports so that calls can be made to a remote host server using normal telephone lines or dedicated telecommunications lines. When alerted 118, the Agent security system needs to be able to find an available serial port 119-122, once it does so it checks to see if a modem is attached 128-129 and tries to initialize it by sending it an initialization string 132. If successful, it checks for a dialtone, then tries to make a quiet call to a remote host server 141. Once the server has been connected, the client machine attempts to initiate a data transaction with the server so it can send its serial number and other data defined to be part of the transaction 151. The server is configured to connect at 2400 bps with no parity, 8 data bits and 1 stop bit. Thus the client matches this configuration. This allows a high connection reliability.

APPENDIX III - DESCRIPTION OF ROUTINES

SNTLINIT:

After this routine has been loaded high into conventional memory 67 and execution has been passed to it, the machine state is saved 68. Conventional memory is the first 640 kilobytes (655,360 bytes) of memory on an Intel 80X86 compatible computer for example. Registers 15 that are affected by this routine are saved on the stack, "saving the machine state". The stack referred to is a LIFO structure, where the LIFO stands for "last in first out". It is where you can temporarily save the contents of CPU registers so that you can restore their initial values.

The microprocessor register AX is used to pass one of three values to the SNTLINIT routine. Depending upon which of the three values are passed to this routine, three different courses of action will be taken. Each course of action describes how the program will initialize itself. To summarize, this routine initializes the Agent security system from either the partition boot sector 55, the operating system boot sector 56 or the input/output module of the operating system 57.

If the microprocessor register AX contains the value 0:

The partition sector 165 is loaded into memory (which has been overwritten on the disc with the boot sector version of the SUBLOADR module). On execution of this code, the SNTLINIT is called.

SUBSTITUTE SHEET

If the microprocessor register AX contains the value 1:

The boot sector 55 of the hard disk (which has been overwritten on the disc with the boot sector version of the SUBLOADR module) is loaded into memory.

5 On execution of this code, the SNTLINIT routine is called.

If the microprocessor register AX contains the value 2:

10 The first sector of IO.SYS/IBMBIO.COM 57 (which has been overwritten on the disk with the IO version of the SUBLOADR module) is loaded into memory.

15 This routine then tests to see if it is in memory already by checking for the 'RPL' signature 69, 84, 96, 108 located at the start of the address for Interrupt 2FH. If it is in memory, this routine exits 77 (to avoid loading more than one copy of the program into memory). If it is not already in memory, then it points (hooks) Interrupt 2FH to an internal routine 71, and does the same with Interrupt EAH 72. It then hooks Interrupt 8 after saving the original Interrupt 8 vector to an internal memory location (internal to the Agent security system).

20 The machine state is restored 74 and the routine exits by jumping to memory location 0000:7C00H for the partition table and boot sector execution paths or 0070:0000H for the IO execution path 75, 76.

SNTLAPI:

25 This API is for use by an external program. It has three functions as follows:

1. Get state of Agent security system. (Checks to see if Agent is already installed.)
2. Set state of Agent security system.
3. Set serial number of system.

30 SWAPINT:

SwapInt stores the existing interrupt vector by replacing the vector for the interrupt number in the CPU register BX with the new vector pointed to by the CPU register pair DS:CX after it stores the current vector at a location pointed to by the register pair DS:DI. If the CPU register

SUBSTITUTE SHEET

DI contains 0 then the vector for the interrupt number contained in the CPU register BX is not stored.

DELAYFUNC:

5

This is a delay function used for hardware timing purposes. This routine is used in FIG. 3F, block 125.

TIMERISR:

10

Interrupt 8h/1Ch is the System Timer Interrupt which executes 18.2 times per second 117 and is used to do the following:

1. Call the old system timer interrupt.
2. Check to see if a communications interrupt is occurring, exiting immediately if so.
3. Save affected CPU registers.
4. Check for an internal state error, exiting immediately if so.
5. Call the state routine.
6. Restore the saved CPU registers.

15

20

ACTIVEROUTINE:

The ActiveRoutine checks to see if the activation period has been exceeded 118. By activation period we mean a period of time that has elapsed since the last valid security call. This period of time is set during the transaction to the server, but is initially set to approximately 7 days.

25

CHECKNEXT PORT:

This is a check for valid serial ports. d involves checking a table of serial port addresses 120 and then testing them to ensure their functionality 122. If a valid serial port cannot be found, a sleep state is entered 125. Upon awakening, this routine is repeated 119.

30

SUBSTITUTE SHEET

DELAYLOOP:

This delay is used for communications delays due to busy signals or no dial-tone and other problems that can affect the communications link.

5

PORTFINDINT:

This procedure calls the previously described CHECKNEXTPORT function 118, 119 in its quest for a valid serial port to initialize. On finding a valid serial port, it stores the ports address, and its corresponding interrupt vector.

10

PORTFIND:

This is a check to see if the serial communications port is in use 123 by dynamically testing the registers in the universal asynchronous receiver - transmitter (UART) that is associated with the current serial port address. Specifically, it tests the Interrupt Enable Register of the UART. This UART register is read into the AL register of the CPU, and if any of the bits are set (logical 1), then the port is in use, otherwise the port is idle. It also tests the interrupt enable bit of the modem control register in the UART. If the bit is not set (logical 1) then the port is idle and available for use.

15

20

Each serial port in the port table 120 is checked until either a valid one is found 123, or the routine goes to sleep 125. If a serial port is found 123, this routine will decide whether or not to initialize the UART using the system BIOS. Interrupt 14H routine, or bypass this routine, programming the UART registers directly. If an error occurs during this process, the routine is exited, and CHECKNEXT PORT is invoked.

25

If the serial port is successfully initialized 128, 129 to the predefined bit rate, parity, word size, number of stop bits etc., the UART is cleared of any pending errors. The serial port buffer is flushed (emptied), so there is no chance of old data being picked up a second time. The state flag that the timer interrupt checks on each clock tick is cleared, as interrupt driven communications have not yet been set up. The appropriate interrupt number is selected and the old interrupt vector is swapped with the new one by calling SWAPINT. The statuses RTS (Request to Send) and DTR (Data Terminal Ready), are enabled in the UART. The 8259 PIC

30

SUBSTITUTE SHEET

is then unmasked, interrupts are enabled in the UART, then the hardware interrupts for the CPU are enabled. Then this routine exits.

MODEMFINDDELAY:

5

This procedure sets the [state-routine] function pointer to point to the MODEMFINDINTT routine, delaying execution until the next interrupt.

MODEMFINDINTT:

10

This routine points to a string to send to the modem, then calls the COMTRANSINTT routine.

MODEMINTTINTT:

15

This procedure tries to initialize the modem 130 with an appropriate initialization string from a table of initialization strings 131, and will try until either the modem is initialized or there are no more initialization strings in the table to try. The COMTRANSINTT routine is called from within this procedure 132-136.

20

MODEMINTT:

This procedure checks the state of the transmission, and checks for incoming data by calling the COMTRANS and COMTRANSCHECK routines 132. This procedure ends by jumping to a jump table which points to the next appropriate routine.

25

MODEMCALLINTT:

30

This routine attempts to place a call 137, 138 by selecting a telephone number 139 (and its appropriate prefix if necessary) from a table of dial strings 140. It will continue to do so until either a call is completed 148 or there are no more initialization strings in the table to try. If a call could not be made 144 then the CLEANUPROUTINE and ERRORROUTINE procedures are to be run during the next state(s) (Interrupt 8 system timer ticks) 155.

SUBSTITUTE SHEET

MODEMCALLINT2:

5 This routine checks the state of the transmission, ending if it is complete. This procedure is called from within the MODEMCALLINIT routine. It in turn calls the MODEMCALL procedure.

MODEMCALL:

10 This routine checks the state of the transmission, ending if it is incomplete. It also checks to see if data has been received yet or not.

MODEMCONNECTINIT:

15 This procedure waits for a query from the host server 148 (at the other end of the communications link), and sends the serial number 151 of the computer. If a call could not be made then the CLEANUPROUTINE and ERRORROUTINE procedures 155 are to be run during the next state(s) (Interrupt 8 system timer ticks).

MODEMCONNECT:

20 This routine checks the state of the transmission, ending if the transmission is incomplete.

CLEANUPROUTINE:

25 This routine resets the Agent security system 155, 156 (sometimes referred to as Sentinel in the source code) back to a known state (ACTIVE), zeroes the transmissionstate flags, flushes the UART buffer. Then it disables all interrupts, restores the old communications interrupt service routine via the SWAPINT procedure. It then sets the state routine function pointer to the CLEANUPROUTINE (to be run during the next Interrupt 8).

30

ERRORROUTINE:

The Agent security system state is set to SNTL STATEERROR (the Agent security system is put in an error state).

SUBSTITUTE SHEET

COMISR:

5 The interrupt service routine used to control one of the systems serial communications ports (and one of the Interrupt Request lines) in order to provide telecommunications services to the Agent security system. It calls the SENDBYTE and BUT PUTCHAR procedures. It handles the low-level details of sending and receiving data during the transmission when it happens.

SENDBYTE:

10 This procedure attempts to send a byte of data to the referenced serial communications port (a variable containing the port address). This routine is used in 141, 151.

COMTRANSINIT:

15 This procedure initializes a transaction between the Agent security system and the modem. A transaction involves sending a string of data 151 to the modem to be sent via telecommunications link to a host server, which after receiving the string of data, in return, sends back a string of data to the client machine 152 containing the Agent security system. The returned string can then be analyzed by the Agent security system to determine what action should be taken next.

20

COMTRANS:

This procedure handles much of the technical details regarding the maintenance of the transaction between the Agent security system and the host server 129, 134, 135, 143, 144, 145, 152, 157. It is primarily responsible for error handling such as incomplete transactions and stalled transmissions.

25

COMTRANSCHECK:

30 Checks the results of a completed transaction between the host server, and the client security system against a table of strings. Three possible outcomes are allowed for:

1. If the incoming data has not been completely received, the carry flag of the client CPU is set (logical 1).

SUBSTITUTE SHEET

2. If the function timed out (exceeded a time threshold value) and no Agent security system internal string matched the string received from the host server, the carry flag of the client CPU is set, and the AH register is zeroed.
- 5 3. If a matching string was found, the carry flag on the client CPU is reset (local O), and the AL register contains a value that matches the internal table entry.

BUF_FLUSH:

- 10 Flushes the internal serial port communications receive buffer on the client machine (containing Agent security system).

15 The buffer is a circular queue. A circular queue is a data structure that has what is called a head pointer and a tail pointer where the head pointer chases the tail pointer around the queue, never really catching it, but processes each byte of the data stored in it. As a byte of data is received by the serial port, it is latched and must be put into a buffer (an area of memory reserved for this purpose) before the next byte arrives (which overwrites the existing latched byte).

20 Whenever a communications session starts, it is important that both the input and output buffers are flushed so that all new incoming and outgoing data are not contaminated by old data still sitting in the buffer.

BUF_GETCHAR:

- 25 Gets a character from the internal serial port communications receive buffer, removing it from the buffers as it does so.

BUF_PUTCHAR:

- 30 Adds a character to the internal serial port communications receive buffer. Increments the head pointer, checking to see if the buffer is full, and setting the carry flag if it is.

SUBSTITUTE SHEET

BUF_INC_PTR:

Increments the receive buffer pointer assigned to the client CPU register SI, and wraps it if necessary.

5

INT2FVECT:

Reserves the required space at the top of conventional memory for the RAM resident portion of the Agent security system. The undocumented Interrupt 21 H, Function 4AH, SubFunction 06 is used to do this.

10

APPENDIX IV - SOURCE CODES

Electronic Article Surveillance System

Source Code for Client-side

(Tazam Assembler Code by Borland)

15

```
*****
*****
; Copyright (c) Absolute Software 1994, 1995
;
20 ; SENTINEL.INC - Sentinel definition file
;
; * PURPOSE:
; *   This file contains or INCLUDEs all constants, macros, and
; *   directives used
25 ; *   by the Sentinel Module.
;
; * HISTORY:
; *   1995.09.05 - CCOTI
; *   New source file taken from build 63a.
30 ; *   See the subdirectory OldFiles for the original
; *   SENTINEL.INC
;
; * NOTES:
;
35 ;*****
;*****

IDEAL
40 parsing mode.                ; Set
JUMPS
local jumps.                  ; Allow
P286N
instructions only.            ; Allow 286

45 INCLUDE "UART.INC"
constants.                    ; RS232 UART

; Enable Debugging.
Debug = 0
```

SUBSTITUTE SHEET

```

; Sentinel Signature.
SNTL_SIG1      = OFDFEh
SNTL_SIG2      = OEPCDh

5  ; Sentinel Version number.
    SNTL_VERSION = 0 * 256 + 100

; Conditional compilation switches.
10  EMIT_ON = 0 ; enables debugging.
    IODELAY_ON = 1 ; enables io delays.
    TWODSKHKS = 0 ; to maintain deflection with 32-
        bit disk access

; Timing & Delays.
15  PORT_LOOP_DELAY = 18 ; 1 second delay
    DIAL_LOOP_DELAY = 18 * 5 ; provide an inter-dial delay of
        5 seconds

20  PREINT13_TIMEOUT = 18 * 120 ; Timeout before sentinel hook
        the system.

; Magic Numbers and Fixed Offsets.
25  DATA_SECTOR_OFFSET = 130h ; MUST Be Sector aligned for disk
        write ; (see Int13ISR)

; Debug macros.
MACRO EMIT ch
30  IF EMIT_ON
        PUSH    AX
        MOV     AL,ch
        CALL    PutChar
        POP     AX
    ENDIF
35  ENDM

MACRO IODELAY
    IF IODELAY_ON
40  CALL DelayFunc
    ENDIF
    ENDM

;*****DO NOT CHANGE WITHOUT UPDATING SUBLOADR.H*****
; Sentinel State constants.
45  SNSTACTIVE      = 0
    SNSTALERT       = 1
    SNSTCALLING     = 2
    SNSTCONNECT     = 3
    SNSTERROR       = 4
50  SNTL_STATE_ERROR ; Check for error: >=

;*****

; Bit flag settings for <xmit state flags>.
55  XMIT_RECEIVE_BIT = 00000001b
    XMIT_SEND_BIT    = 00000010b
    XMIT_SENT_AWK_BIT = 00000100b
    XMIT_RECEIVE_AWK_BIT = 00001000b

60  IFDEF Testing
        RECEIVE_TIMEOUT = OFFFh ; test timeout huge
    ELSE
        RECEIVE_TIMEOUT = 18 * 40 ; timeout == 40 seconds.
    ENDIF
```

SUBSTITUTE SHEET

```

                    ; timer values (based on 18
ticks/second)
5   TM1SEC          = 18 * 1
    TM2SEC          = 18 * 2
    TM3SEC          = 18 * 3
    TM4SEC          = 18 * 4
    TM5SEC          = 18 * 5
    TM6SEC          = 18 * 5
    TM2SEC          = 18 * 5
10  TM10SEC         = 18 * 10
    TM30SEC         = 18 * 30
    TM40SEC         = 18 * 40
    TM1MIN          = 18 * 60
    TM2MIN          = 18 * 60 * 2
15
    SNMDMFINDTO     = 18 * 5           ; timeouts
    seconds         ; modem find timeout -5
    SNMDHINITTO     = 18 * 5           ; modem initialization timeout
    -5 seconds      ;
20  SNMDMDLTO       = 18 * 40         ; modem dial out timeout -40
    seconds         ;
    SNRESPONSETO    = 18 * 40         ; server response timeout -40
    seconds         ;
25  SNPWRUPDLYTO    = 18 * 10         ; power-up delay before
    hooking int 2F -10 seconds

30  SNCALLNA        = 0               ; call status
    SNPRTSRCH       = 1               ; no attempt yet
    port            ; searching for an available
    SNMDMSRCH       = 2               ; searching fo a modem on the
    port            ;
35  SNMDMINIT       = 3               ; initializing modem
    SNMDMPD         = 4               ; sending predial string to
    modem           ;
    SNMDMDL         = 5               ; sending dial string to modem
    SNWTCON         = 6               ; waiting for modem to connect
    to server       ;
40  SNWTENQ         = 7               ; waiting for ENQ from server
    SNWTACK         = 8               ; waiting for ACK from server
    SNWTNCD         = 9               ; waiting for next-call-date
    from server     ;
45  SNCALLPASS      = 10              ; call passed
    SNCALLFAIL      = 11              ; call failed

50  STRUC RXZCM      ; receiver structure
    rxstate         DW ?              ; receiver state
    rxxtmr          DW ?              ; receive timer
    rxxlrc          DB ?              ; received packet running-sum
    LRC
    rxpktlen        DW ?              ; packet length to receive
    rxbytcnt        DW ?              ; received bytes in current
    packet           ;
55  rxstype         DB ?              ; packet type
    rxstype         DB ?              ; packet subtype
    rxbufp          DW BYTE PTR ?     ; pointer to receive buffer
    ENDS RXZCM

60  STRUC TXZCM      ; transmit structure
    txstate         DW ?              ; current transmitter state
    txxtmr          DW ?              ; next transmitter state
    txxtmr          DW ?              ; transmit timer
    txpkttyp        DB ?              ; packet type to transmit

```

SUBSTITUTE SHEET

```

    txxtxing      DB 0                      ; transmission in progress
flag
    txxnakcnt     DB 0                      ; transmit NAK count
    txxenqcnt     DB 0                      ; transmit ENQ count
5    txxlrc        DB ?                      ; transmit packet running-sum
LRC
    txxpktlen     DW ?                      ; remaining data bytes to
transmit
    txxdatcnt     DW ?                      ; index of next data byte to
10    tranemit
    txxttype      DB ?                      ; packet type
    txxsotype     DB ?                      ; packet subtype
    txxbuff       DW BYTE PTR ?            ; pointer to transmit buffer
15    ENDS TXZCM

                                ; transmit packet types:
                                ;   data packet
                                ;   modem packet
20    CMTXDATPKT   = 0
    CMTXMDMPKT    = 1
    CMTXDLACK     = 2
    CMTXDLNAX     = 3
    CMTXDLENQ     = 4
    CMTXDLEOT     = 5
                                ;   datalink ACK
                                ;   datalink NAK
                                ;   datalink ENQ
                                ;   datalink EOT

25    DLSTX       = 2h
    DLETX        = 3h
    DLEOT        = 4h
    DLENQ        = 5h
    DLACK        = 6h
30    DLNAX       = 15h
                                ; protocol control characters
                                ; STX character
                                ; ETX character
                                ; EOT character
                                ; ENQ character
                                ; ACK character
                                ; NAK character

                                ; protocol message types
                                ; message from the server
35    SNSERVER     = 80h

                                ; protocol message subtypes
                                ; next call packet
    SNNEXTCALL    = 0h
    SNDISABLE     = 1h
                                ; disable sentinel packet

40    SNSNTLSIZE   = 11
                                ; Sentinel sector size
```

SUBSTITUTE SHEET

```
*****
*****
; * Copyright (c) Absolute Software 1994, 1995
; *
5 ; * SNTLAPI.INC
; *
; * Contains global labels for the api module.
; *
10 ; * HISTORY:
; *   1995.09.05 - CCOTI
; *               Created.
; *
; *****
; *****
15
SEGMENT SNTL_SEG BYTE PUBLIC 'CODE'

    GLOBAL SntlAPI                : FAR
20    GLOBAL SwapInt              : NEAR

    IF IODELAY ON
        GLOBAL DelayFunc          : NEAR
25    ENDIF

    GLOBAL CmpDates              : NEAR
30    ENDS
```

SUBSTITUTE SHEET

```
5  ;*****
   ;*****
   ;* Copyright (c) Absolute Software 1994, 1995
   ;*
   ;* SNTLBUFF.INC
   ;*
   ;* Contains global labels for the buffer module.
   ;*
10  ;* HISTORY:
   ;*   1995.09.05 - CCOTI
   ;*               Created.
   ;*
   ;*****
15  ;*****

   SEGMENT SNTL_SEG BYTE PUBLIC 'CODE'

   GLOBAL buf_flush           : NEAR
20  GLOBAL buf_getchar        : NEAR
   GLOBAL buf_putchar         : NEAR
   GLOBAL buf_inc_ptr         : NEAR

ENDS
```

SUBSTITUTE SHEET

```
*****
*****
5  ; * Copyright (c) Absolute Software 1994, 1995
   ; *
   ; * SNTLCOMM.INC
   ; *
   ; * Contains the global labels for the comm module.
10 ; * HISTORY:
   ; *   1995.09.05 - CCOTI
   ; *               Created.
   ; *
   ; *
   ; *
15 *****
   *****

   SEGMENT SNTL_SEG BYTE PUBLIC 'CODE'

20   GLOBAL cmftxnak           : NEAR
   GLOBAL cmftxenq           : NEAR
   GLOBAL cmfprpmdm          : NEAR
   GLOBAL cmftx              : NEAR
   GLOBAL cmfpack            : NEAR

25   ENDS
```

SUBSTITUTE SHEET


```

;*****
;*****
5  ;* Copyright (c) Absolute Software 1994, 1995
;*
;* SNTLCOMV.INC
;*
;* Contains global lable for the Comm ISR.
;*
10 ;* HISTORY:
;*   1995.09.05 - CCOTI
;*               Created.
;*
;*****
;*****
15
SEGMENT SNTL_SEG BYTE PUBLIC 'CODE'
    GLOBAL cmfisar
    : FAR
20 ENDS
```

SUBSTITUTE SHEET

```
;*****
*****
;* Copyright (c) Absolute Software 1994, 1995
;*
5  ;* SNTLDATA.INC
;*
;* PURPOSE:
;*   Contains the global labels for the data segment.
10 ;* HISTORY:
;*   1995.09.05 - CCOTI
;*               Created.
;*
15 ;*****
*****

SEGMENT SNTL_SEG BYTE PUBLIC 'CODE'

20     GLOBAL engstftn          : WORD
        GLOBAL Sentinel_state  : BYTE

        ;Scatch vars to store the current port info being used.
        GLOBAL engmdmprt       : WORD
        GLOBAL engmdmprtint    : WORD
25     GLOBAL engmdmprtadd      : WORD

        ;Previous ISR vectors.
        GLOBAL engprvtmr       : DWORD
        GLOBAL engprvcom       : DWORD
30     GLOBAL engprvdsk1       : DWORD

        IF TWODSKHKS
        GLOBAL engprvdsk2      : DWORD
        GLOBAL eng2dskhks      : BYTE
35     GLOBAL engdskskip       : BYTE
        ENDIF

        GLOBAL engprvint2f     : DWORD

40     ;ROR'd to limit updating the real-time clock once every 16 ticks (see
        ;ActiveRoutine).
        GLOBAL cycle_var      : WORD

        GLOBAL win_flag       : BYTE
45     GLOBAL win_vm          : BYTE

        GLOBAL engincmiser    : BYTE

        GLOBAL send_buf_len   : WORD
50     GLOBAL send_buf_ptr    : WORD

        GLOBAL engcomcnt      : WORD
        GLOBAL engcomerr      : BYTE
        GLOBAL TimerISR_count : WORD
55     GLOBAL sent_count      : WORD
        GLOBAL received_count : WORD
        GLOBAL engflcnt       : BYTE
        GLOBAL engclst        : BYTE
        GLOBAL engcomhk       : BYTE
60     GLOBAL engsuspend      : BYTE
        GLOBAL engdlytmr      : WORD
        GLOBAL engint2ftmr     : WORD
        GLOBAL engprtdlytmr   : WORD
        GLOBAL engdeflect     : BYTE
65     GLOBAL dkgcyl          : WORD
```

SUBSTITUTE SHEET

```
GLOBAL dkgsctr          : BYTE
GLOBAL sngapifl         : BYTE
GLOBAL sngpwd1          : WORD
5 GLOBAL sngpwd2          : WORD

;Sentienl Settings.

GLOBAL modem_default_port : WORD
10 GLOBAL port_table      : WORD
   PORT_TABLE_SIZE = 4

; Disk location of data sector.
15 GLOBAL data_cyl_sect   : WORD
   GLOBAL data_head_drive : WORD
   GLOBAL sngdskwrt       : BYTE

; Output strings.
20 GLOBAL init_str_num    : WORD
   GLOBAL init_str_table : WORD : 5
   INIT_STR_TABLE_SIZE = 6

25 GLOBAL dial_str_num    : WORD
   GLOBAL dial_str_table : WORD : 4
   DIAL_STR_TABLE_SIZE = 5

30 GLOBAL dial_number     : BYTE

GLOBAL sn_packet_start   : UNKNOWN
GLOBAL stx_byte          : BYTE
GLOBAL lsb_length_byte   : BYTE
35 GLOBAL msb_length_byte : BYTE
GLOBAL sn_text_start     : UNKNOWN
GLOBAL text_type         : BYTE
GLOBAL text_sub_type     : BYTE
GLOBAL sn_data_start     : UNKNOWN
40 GLOBAL sngsernum       : BYTE : 6
GLOBAL now_date          : UNKNOWN
GLOBAL now_year          : BYTE
GLOBAL now_month         : BYTE
GLOBAL now_day           : BYTE
45 GLOBAL now_hour        : BYTE
GLOBAL now_minute       : BYTE
GLOBAL sn_data_end       : UNKNOWN
GLOBAL etx_byte          : BYTE
GLOBAL lrc_byte          : BYTE
50 GLOBAL sn_packet_end   : UNKNOWN
GLOBAL sngsernum_str     : UNKNOWN
GLOBAL sngsernum_str_len : BYTE
GLOBAL sngdatalen        : BYTE

55 GLOBAL rx              : RXZCM

GLOBAL tx                : TXZCM

; Result tables.
60 GLOBAL command_result_table_len : BYTE
   GLOBAL command_result_table    : UNKNOWN

GLOBAL mdm_init_result_table_len : BYTE
GLOBAL mdm_init_result_table     : UNKNOWN
65 GLOBAL dial_result_table_len   : BYTE
```

SUBSTITUTE SHEET

```
GLOBAL dial_result_table          : UNKNOWN
GLOBAL connect_result_table_len   : BYTE
5 GLOBAL connect_result_table      : UNKNOWN
; Modem and result string pool.
GLOBAL string_pool                 : BYTE : 127
10 GLOBAL modem_find_str           : UNKNOWN
; next call date
GLOBAL next_call_date              : UNKNOWN
GLOBAL next_call_year              : BYTE
15 GLOBAL next_call_month          : BYTE
GLOBAL next_call_day               : BYTE
GLOBAL next_call_hour              : BYTE
GLOBAL next_call_minute            : BYTE
20 GLOBAL sngrxbufhd               : WORD
GLOBAL sngrxbufst1                 : WORD
GLOBAL sngrxbufst                  : UNKNOWN
GLOBAL sngrxbuf                    : BYTE
GLOBAL sngrxbufend                 : UNKNOWN
25 GLOBAL nextcall_text            : BYTE : 5
GLOBAL sngtxindex                  : BYTE
GLOBAL sngtxbufst                  : UNKNOWN
30 GLOBAL sngtxbuf                 : BYTE
GLOBAL sngtxbufend                 : UNKNOWN
; Result jump tables.
35 ; Table for ModemFind
GLOBAL find_jump_table             : CODEPTR
; Table for ModemInit.
40 GLOBAL init_jump_table          : CODEPTR
; Table for dial results.
GLOBAL dial_jump_table             : CODEPTR
45 GLOBAL cnct_jump_table          : CODEPTR
ENDS
```

SUBSTITUTE SHEET

```
*****
*****
5  ; * Copyright (c) Absolute Software 1994, 1995
   ; *
   ; * SNTLI13V.INC
   ; *
   ; * PURPOSE:
10  ; *   Contains INT 13 ISRs and disk deflection routines.
   ; *
   ; * HISTORY:
   ; *   1995.09.05 - CCOTI
   ; *               Created.
   ; *
15  ;*****
   ;*****

   SEGMENT SNTL_SEG BYTE PUBLIC 'CODE'

20  GLOBAL load_time      : WORD
   GLOBAL Int13ISR       : FAR

ENDS
```

SUBSTITUTE SHEET

```
*****
*****
; * Copyright (c) Absolute Software 1994, 1995
; *
5 ; * SNTLI2FV.INC - SNTLI2FV.ASM global lables.
; *
; * PURPOSE:
; *
; * HISTORY:
10 ; *   1995.09.05 - CCOTI
; *               Created.
; *
; * NOTES:
; *
15 ; *
; *
*****

SEGMENT SNTL_SEG BYTE PUBLIC 'CODE'

20     GLOBAL Int2FVect      : FAR
       GLOBAL snfint2f      : FAR

ENDS
```

SUBSTITUTE SHEET

```
*****
*****
5  ; * Copyright (c) Absolute Software 1994, 1995
   ; *
   ; * SNTLJTBL.INC
   ; *
   ; * Contains the global labels for the jump table.
   ; *
10  ; * HISTORY:
   ; *   1995.09.05 - CCOTI
   ; *               Created.
   ; *
   ; *****
   ; *****
15
   SEGMENT SNTL_SEG BYTE PUBLIC 'CODE'

   GLOBAL JumpTable                : NEAR
20  GLOBAL cleanup                  : NEAR
   GLOBAL find_ok                   : NEAR
   GLOBAL find_timeout              : NEAR
25  GLOBAL init_ok                  : NEAR
   GLOBAL init_error                : NEAR
   GLOBAL dial_server               : NEAR
30  GLOBAL dial_busy                : NEAR
   GLOBAL dial_error                : NEAR
   GLOBAL dial_no_carr              : NEAR
   GLOBAL dial_no_tone              : NEAR
35  GLOBAL cnct_ack                 : NEAR
   GLOBAL cnct_enq                  : NEAR
   GLOBAL cnct_error                : NEAR
   GLOBAL cnct_eot                  : NEAR
40  GLOBAL cnct_nak                 : NEAR
   GLOBAL cnct_resend               : NEAR
   GLOBAL cmrxpktto                : NEAR

ENDS
```

SUBSTITUTE SHEET

```
5  ;*****
   ;*****
   ;* Copyright (c) Absolute Software 1994, 1995
   ;*
   ;* SNTLSTRT.INC
   ;*
   ;* Contains global labels for the string table module.
   ;*
10  ;* HISTORY:
   ;*   1995.09.05 - CCOTI
   ;*               Created.
   ;*
   ;*****
15  ;*****
   SEGMENT SNTL_SEG BYTE PUBLIC 'CODE'
       GLOBAL ComTransCheck
20  ENDS                                : NEAR
```

SUBSTITUTE SHEET


```
*****
*****
; * Copyright (c) Absolute Software 1994, 1995
; *
5 ; * SNTLTIMR.ASM
; *
; * Contains the global labels for the TimerISR.
; *
10 ; * HISTORY:
; *   1995.09.05 - CCOTI
; *               Created.
; *
*****
*****
15
SEGMENT SNTL_SEG BYTE PUBLIC 'CODE'

GLOBAL tmf_isr                : FAR
GLOBAL ActiveRoutine          : NEAR
20 GLOBAL snfsrst              : NEAR
GLOBAL ModemInitInit          : NEAR
GLOBAL ModemCallInit          : NEAR
GLOBAL ModemFindInit          : NEAR
25 GLOBAL snftxchkin           : NEAR
GLOBAL snfgetpkt              : NEAR

ENDS
```

SUBSTITUTE SHEET

```
;
; UART.INC -- Asm header file for programming the UART chip.
;

5  ; UART memory port base addresses
    COM1_ADDRESS equ 3F8h
    COM2_ADDRESS equ 2F8h
    COM3_ADDRESS equ 3E8h
10  COM4_ADDRESS equ 2E8h

    ; UART port interrupts
    COM1_INTERRUPT equ 04h
    COM2_INTERRUPT equ 03h
15  COM3_INTERRUPT equ 04h
    COM4_INTERRUPT equ 03h

    ; UART memory port offsets
    THR equ 0 ; Transmitter holding register (out).
    RDR equ 0 ; Receiver data register (in).
20  BRDL equ 0 ; Low byte, baud rate divisor (alternate
port).
    IER equ 1 ; Interrupt enable register.
    BRDH equ 1 ; High byte, baud rate divisor (alternate
25  port).
    IIR equ 2 ; Interrupt ID register.
    LCR equ 3 ; Line control register.
    MCR equ 4 ; Modem control register.
    LSR equ 5 ; Line status register.
30  MSR equ 6 ; Modem status register.

    ; UART memory bit masks

    ; Interrupt enable register.
35  IER_RDR_FULL equ 00000001b
    IER_THR_EMPTY equ 00000010b
    IER_DATA_ERR equ 00000100b
    IER_MSR_CHANGED equ 00001000b

    ; Interrupt ID register.
40  IIR_MULT_INT equ 00000001b

    IIR_INT_ID_MASK equ 00000110b
    IIR_MSR_CHANGED equ 00000000b
    IIR_THR_EMPTY equ 00000010b
45  IIR_RDR_FULL equ 00000100b
    IIR_DATA_ERR equ 00000110b

    ; Line control register.
50  LCR_CHAR_MASK equ 00000011b
    LCR_CHAR_5 equ 00000000b
    LCR_CHAR_6 equ 00000001b
    LCR_CHAR_7 equ 00000010b
    LCR_CHAR_8 equ 00000011b

55  LCR_STOP_BIT_MASK equ 00000100b
    LCR_1STOP_BIT equ 00000000b
    LCR_2STOP_BIT equ 00000100b

    LCR_PARITY_MASK equ 00111000b
60  LCR_NO_PARITY equ 00000000b
    LCR_ODD_PARITY equ 00100000b
    LCR_EVEN_PARITY equ 00110000b
    LCR_MARK_PARITY equ 00101000b
65  LCR_SPACE_PARITY equ 00111000b
```

SUBSTITUTE SHEET

- 45 -

```

LCR_BREAK_MASK      equ 01000000b
LCR_BREAK_OFF       equ 00000000b
LCR_BREAK_ON        equ 01000000b

5   LCR_PORT_MASK    equ 10000000b
    LCR_NORMAL_PORT  equ 00000000b
    LCR_ALT_PORT     equ 10000000b

10  ; Modem control register.
    MCR_DTR_ON       equ 00000001b
    MCR_RTS_ON       equ 00000010b ; NOT CONFIRMED!!!
    MCR_USER_OUT_1    equ 00000100b
    MCR_ENABLE_INT    equ 00001000b
    MCR_UART_TEST     equ 00010000b

15  ; Line status register.
    LSR_RDR_FULL      equ 00000001b
    LSR_OVER_ERR      equ 00000010b
    LSR_PARITY_ERR     equ 00000100b
    LSR_FRAMING_ERR    equ 00001000b
    LSR_BREAK         equ 00010000b
    LSR_THR_EMPTY     equ 00100000b
    LSR_TSR_EMPTY     equ 01000000b

25  ; Modem status register.
    MSR_CTS_CHANGED   equ 00000001b
    MSR_DSR_CHANGED   equ 00000010b
    MSR_RI_CHANGED    equ 00000100b
    MSR_DCD_CHANGED    equ 00001000b
    MSR_CTR_ACTIVE     equ 00010000b
    MSR_DSR_ACTIVE     equ 00100000b
    MSR_RI_ACTIVE      equ 01000000b
    MSR_DCD_ACTIVE     equ 10000000b

35  ; BIOS services.
    BIOS_INIT_PORT    equ 00h
    BIOS_WRITE_PORT   equ 01h
    BIOS_READ_PORT    equ 02h
    BIOS_STATUS_PORT  equ 03h

40  ; BIOS initialization values.
    BIOS_7BITS        equ 00000010b
    BIOS_8BITS         equ 00000011b
    BIOS_1STOP         equ 00000000b
    BIOS_2STOP         equ 00000100b

    BIOS_PARITY_MASK   equ 00011000b
    BIOS_NO_PARITY     equ 00000000b
    BIOS_ODD_PARITY    equ 00001000b
    BIOS_EVEN_PARITY   equ 00011000b

50  BIOS_BAUD_MASK     equ 11100000b
    BIOS_110_BAUD      equ 00000000b
    BIOS_150_BAUD      equ 00100000b
    BIOS_300_BAUD      equ 01000000b
    BIOS_600_BAUD      equ 01100000b
    BIOS_1200_BAUD     equ 10000000b
    BIOS_2400_BAUD     equ 10100000b
    BIOS_4800_BAUD     equ 11000000b
    BIOS_9600_BAUD     equ 11100000b

60

```

SUBSTITUTE SHEET

```

;*****
;*****
;* Copyright (c) Absolute Software 1994, 1995
;*
5  ;* SENTINEL.ASM - Sentinel Initialization and TSR Code
;*
;* PURPOSE:
;*   This is the main build file for the Sentinel module.
10 ;* HISTORY:
;*   1995.09.05 - CCOTI
;*   Source taken from build 63a and broken up into
;*   separate source
15 ;*   files. See the subdirectory OldFiles for the original
;*   SNTLINIT.ASM
;*
;*****
;*****

20  IDEAL
    include "SENTINEL.INC"

;...
;...
25 ;*****
;*****
;*
;* SNTL_SEG - Resident segment.
;*
30 ;*****
;*****
    SEGMENT SNTL_SEG PARA PUBLIC 'CODE'

;*****
;*****
35 include "SNTLJTBL.ASM"

;*****
;*****
40 include "SNTLCONV.ASM"

;*****
;*****
45 include "SNTLSTRT.ASM"

;*****
;*****
50 include "SNTLBUFP.ASM"

;*****
;*****
55 include "SNTLI2FV.ASM"

;*****
;*****
    ENDS

60      END
```

SUBSTITUTE SHEET

```
*****
*****
; * Copyright (c) Absolute Software 1994, 1995
; *
5  ; * SNTLAPI.ASM
; *
; * Contains the sentinel API routine and general purpose routines
; * used by all
10 ; * modules.
; *
; * HISTORY:
; *   1995.09.05 - CCOTI
; *             Created.
; *
15 ; *
*****

IDEAL

20  %NOLIST
    include "SENTINEL.INC"
    include "SNTLAPI.INC"
    include "SNTLDATA.INC"
    include "SNTLTIMR.INC"
25  %LIST

    SEGMENT SNTL_SEG BYTE PUBLIC 'CODE'

30  ; *
    ; *
    ; * SNTLAPI
    ; *
    ; * PURPOSE:
35  ; *   This function provides an external API for the Ward and Tender
    ; *   modules,
    ; *   as well as development software tools, to gain access to the
    ; *   Sentinel.
    ; *
    ; *   The following functions are supported:
    ; *
    ; *   Function 0 - Get Sentinel State
    ; *       returns AL = Sentinel_state
    ; *               BX = sngstftn
45  ; *
    ; *   Function 1 - Set Sentinel State to ALERT
    ; *       returns CF = 0 if successful
    ; *               CF = 1 if failed
    ; *
    ; *   Function 2 - Get Sentinel Version Number
    ; *       returns AH = major version number
    ; *               AL = minor version number
50  ; *
    ; *   Function 3 - Get Sentinel Serial Number
    ; *       returns ES:DI = pointer to serial number
55  ; *
    ; *   Function 4 - Cancel Sentinel ALERT
    ; *       returns CF = 0 if successful
    ; *               CF = 1 if failed
60  ; *
    ; *   Function 5 - Set next-call date and time
    ; *       returns ES = Sentinel data segment
    ; *               DI = offset of next_call_date
    ; *               SI = offset of sngdskwrt
65  ; *
```

SUBSTITUTE SHEET

```

5      ;*      Function 6 - Get call status
      ;*      returns AL = sngclst: SNCALLNA   = 0  no call attempt yet
      ;*      SNPRTSRCH   = 1  searching for an
      ;*      available port
      ;*      SNMDMSRCH   = 2  searching fo a modem
      ;*      on the port
      ;*      SNMDMINIT   = 3  initializing modem
      ;*      SNMDMPD     = 4  sending predial string
10     ;*      to modem
      ;*      SNMDMDL     = 5  sending dial string to
      ;*      modem
      ;*      connect to server
      ;*      SNWTCON     = 6  waiting for modem to
15     ;*      server
      ;*      SNWTENQ     = 7  waiting for ENQ from
      ;*      server
      ;*      SNWTACK     = 8  waiting for ACK from
      ;*      SNWTNCD     = 9  waiting for next-call-
20     ;*      date from server
      ;*      SNCALLPASS = 10 call passed
      ;*      SNCALLFAIL = 11 call failed
      ;*
      ;*      Function 7 - Disable Sentinel disk deflection
25     ;*      returns CF = 0 if successful
      ;*      CF = 1 if failed
      ;*
      ;*      Function 8 - Enable Sentinel disk deflection
30     ;*      returns CF = 0 if successful
      ;*      CF = 1 if failed
      ;*
      ;*      Function 9 - return data segment pointers
      ;*      returns ES:DI = Sentinel Data Segment (SntlDataSeg in
35     ;*      sentinel.h)
      ;*      ES:SI = Sentinel Settings (SntlSettings in
      ;*      sentinel.h)
      ;*
      ;*      PARAMETERS:
40     ;*      None
      ;*
      ;*      Registers destroyed: none
      ;*
      ;*      Globals referenced:
45     ;*      Sentinel_state
      ;*
      ;*      Globals modified:
      ;*      Sentinel_state - set to SNSTALERT by function 1
      ;*      sngstftn - set to
50     ;*
      ;*      BIOS calls: none
      ;*
      ;*      DOS calls: none
      ;*
      ;*      proc calls: none
55     ;*
      ;*      hardware access: none
      ;*
      ;*      *****
      ;*      *****
60     ASSUME CS:SNL_SEG, DS:NOTHING, ES:NOTHING
      PROC SntlAPI FAR
      @@check0:
      ; Return the state.
      CMP     AH,0
      JNE     @@check1
65     MOV     AL,[Sentinel_state]

```

SUBSTITUTE SHEET

```

        MOV     BX,[sngstftn]
        RET

5  @@check1:
   to ALERT.                                ; Attempt to set the state
        CMP     AH,1
        JNE     @@check2
        CMP     [Sentinel_state],SNSTACTIVE
        JNE     @@exit_w_error
10  MOV     [Sentinel_state],SNSTALERT
   ; MOV     [sngstftn], OFFSET snfsenrst
        CLC
        RET

15  @@check2:
   number.                                ; Return the version
        CMP     AH,2
        JNE     @@check3
        MOV     AX,SNTL_VERSION
20  RET                                ; MOD CCOTI 48:95.01.27

@@check3:
        CMP     AH,3                                ; Return the serial number.
        JNE     @@check4
25  PUSH     CS
        POP     ES
        MOV     DI,OFFSET sngsernum
        RET

30  @@check4:
        CMP     AH,4
        JNE     @@check5
        CMP     [Sentinel_state], SNSTACTIVE
        JE      @@check4_done
35  MOV     [Sentinel_state], SNSTACTIVE
        MOV     [sngstftn], OFFSET snfsenrst
@@check4_done:
        RET

40  @@check5:
        CMP     AH,5                                ; test for function 5
        JNE     @@check6                                ; not detected, continue

45  PUSH     CS                                ; prepare to copy string
        POP     ES                                ; get ES = CS
        next_call_date
        MOV     DI, OFFSET next_call_date            ; ES:DI points to
50  data_write flag
        MOV     SI, OFFSET sngdskwrt
        RET                                ; ES:SI points to

60  RET                                ; exit

55  @@check6:
        CMP     AH,6                                ; test for function 6
        JNE     @@check7                                ; not detected, continue

        MOV     AL, [sngclst]
60  AL
        RET                                ; get the call status into
        RET                                ; exit

@@check7:
        CMP     AH, 7                                ; test for function 7
65  JNE     @@check8                                ; not detected, continue

```

SUBSTITUTE SHEET

```

MOV      [sngdeflect], 0          ; clear the Sentinel disk
deflection flag
CLC
RET                                     ; clear the carry flag
; exit
5
@@check8:
CMP      AH, 8                    ; test for function 8
JNE      @@check9                ; not detected, exit with
error
10      MOV      [sngdeflect], 1    ; set the Sentinel disk
deflection flag
;This is commented out to maintain the data segment offset with the
CTM.EXE (See CCOTI).
;      CLC
15      RET                        ; clear the carry flag
; exit

@@check9:
CMP      AH, 9                    ; test for function 9
JNE      @@exit_w_error          ; not detected, exit with
error
20      PUSH     CS
POP      ES                        ; get ES = CS
; ES:DI points to data
segment
MOV      DI, OFFSET sngstftn
25      ; ES:SI points to sentinel
settings
MOV      SI, OFFSET modem_default_port
CLC
RET                                     ; clear the carry flag
30      ; exit

@@exit_w_error:
STC

35      @@exit:
RET

ENDP SntlAPI
ASSUME NOTHING

40      ;*****
;Routine: SwapInt
;
;Descript: SwapInt stores the existing vector
45      ; replaces the vector for the interrupt in BX with the new vector
DS:DX after
; it stores the current vector at [DS:DI]. If DI = 0 the current
vector is
; not stored.
50      ;
;Arguments:
; BX = the interrupt to hook into
; DS:DI = address to save the existing vector; if DI = 0 the
existing vector
55      ; if not stored.
; DS:DX = the new vector to install
;
;Registers destroyed: AX, BX, ES, FLAGS
60      ;
;Returns: nothing
;
;BIOS calls: none
;
65      ;DOS calls: none
;

```

SUBSTITUTE SHEET


```

;proc calls: none
;*****
5  PROC SwapInt
    ASSUME CS:SNTL_SEG, DS:NOTHING, ES:NOTHING
    XOR     AX,AX
    MOV     ES,AX
    SHL     BX,2
    address of vector
10  ; load the existing vector and save it to DS:DI (if requested).
    OR      DI,DI
    JZ      @@no_store
    MOV     AX,[ES:BX]
    MOV     [DS:DI],AX
15  MOV     AX,[ES:BX+2]
    MOV     [DS:DI+2],AX
    @@no_store:
    ; install the new vector
    CLI
20  MOV     [ES:BX],CX
    MOV     [ES:BX+2],DS
    STI
    RETN
25  ENDP SwapInt
    ASSUME NOTHING

;*****
;Routine: DelayFunc
;
;Descript: DelayFunc - introduces an delay.
;
;Arguments: none
;
35  ;Registers destroyed: none
;
;Returns: nothing
;
;BIOS calls: none
40  ;
;DOS calls: none
;
;proc calls: none
;*****
45  IF IODELAY ON
    ASSUME CS:SNTL_SEG, DS:NOTHING, ES:NOTHING
    PROC DelayFunc
50  PUSH     CX
    MOV     CX,1
    @@loop_start:
    LOOP    @@loop_start
    POP     CX
    RETN
55  ENDP DelayFunc
    ASSUME NOTHING
    ENDIF

;*****
60  ;Routine: CmpDates
;
;Descript: CmpDates - compares two dates and sets the CF=1 if date1 <
65  date2.
;

```

SUBSTITUTE SHEET

```

;Arguments: {SI} -> date1
;           {DI} -> date2
;Registers destroyed: SI, DI, CX, ES
5 ;Returns: CF = 1 if date1 < date2
;
;BIOS calls: none
;
10 ;DOS calls: none
;
;proc calls: none
;*****
;*****
15     ASSUME CS:NTL_SEG, DS:NOTHING, ES:NOTHING
PROC CmpDates
    PUSH    DS
    POP     ES
    CLD
    MOV     CX,5
20 @@cmp_loop:
    CMPSB
    JB      @@cmp_exit      ; CMP {SI},{DI}
    LOOPE   @@cmp_loop      ; CF = 1?
25 @@cmp_exit:
    RETN
ENDP CmpDates

    ASSUME NOTHING
30 ENDS

    END
```

SUBSTITUTE SHEET

```

;*****
;*****
; * Copyright (c) Absolute Software 1994, 1995
; *
5  ; * SNTLBUFF.ASM
; *
; * Contains the circular buffer access routines.
; *
10 ; * HISTORY:
; *   1995.09.05 - CCOTI
; *               Created.
; *
;*****
;*****
15
IDEAL

%NOLIST
include "SENTINEL.INC"
20 include "SNTLBUFF.INC"
include "SNTLDATA.INC"
%LIST

25 SEGMENT SNTL_SEG BYTE PUBLIC 'CODE'
    ASSUME CS:SNTL_SEG, DS:SNTL_SEG, ES:NOTHING

;*****
;*****
30 ;
; BUF_FLUSH - flush receive buffer
;
; PURPOSE:
;   This function flushes the receive buffer by setting the tail
index equal
35 ;   to the head index.
;
; PARAMETERS:
;   None
;
40 ; RETURNS:
;   Nothing
;
; REGISTERS DESTROYED:
;   None
45 ;
; GLOBALS REFERENCES:
;   None
;
; GLOBALS MODIFIED:
50 ;   sngrxbufhd
;   sngrxbuftl
;
; BIOS CALLS:
;   None
55 ;
; DOS CALLS:
;   None
;
; PROCEDURE CALLS:
60 ;   None
;
; HARDWARE ACCESS:
;   None
;
65 ; NOTES:
```

SUBSTITUTE SHEET

```

;
;*****
;*****
5  PROC buf_flush NEAR
      MOV     [sngrxbufhd],OFFSET sngrxbuf
      MOV     [sngrxbuftl],OFFSET sngrxbuf
      RET
      ; exit
10  ENDP buf_flush
;*****
;*****
;
;  BUF_GETCHAR - get a character from receive buffer
15  ;
;  PURPOSE:
;      This function returns the next available character in the
;      receive buffer
;      and increments the tail pointer.
20  ;
;Arguments: none
;
;Registers destroyed: AL, SI
;
25  ;Globals referenced:
;      sngrxbuftl
;      sngrxbufhd
;
;Globals modified:
30  ;      received_buf_tail - moved to the location of the next character
;
;Returns: AL = the character taken, CF=0
;          If the buffer is empty CF=1
;
35  ;BIOS calls: none
;
;DOS calls: none
;
40  ;proc calls: buf_inc_ptr
;
;hardware access: none
;*****
;*****
45  PROC buf_getchar NEAR
      MOV     SI, [sngrxbuftl]      ; get the tail pointer
      CMP     SI, [sngrxbufhd]      ; is it the same as the head
      JE      @@empty              ; yes, exit with status
50      MOV     AL,[SI]              ; no, get the next byte
      CALL    buf_inc_ptr           ; increment tail pointer
      MOV     [sngrxbuftl], SI      ; set new tail pointer
position
55      CLC                          ; set status
      RET                          ; exit
@@empty:
60      STC                          ; set status
      RET                          ; exit
ENDP buf_getchar
;*****
;*****
65

```

SUBSTITUTE SHEET

```

;Routine: buf_putchar
;
;Descript: Adds a character to the buffer.
5 ;Arguments: AL = the character to add
;
;Registers destroyed: SI
;
10 ;Globals referenced:
;     sngrxbufhl
;     sngrxbufhd
;
;Globals modified:
15 ;     received_buf_head - moved to the location of the next free
;     space
;
;Returns: CF=0 if the character is stored correctly.
;         CF=1 if the buffer is full.
;
20 ;BIOS calls: none
;
;DOS calls: none
;
25 ;proc calls: buf_inc_ptr
;
;hardware access: none
;*****
;*****
30 PROC buf_putchar NEAR
    MOV     SI, [sngrxbufhd]           ; point to the head of the
buffer
    MOV     [SI], AL                  ; store the received character
35     CALL  buf_inc_ptr               ; increment the head
    MOV     [sngrxbufhd], SI          ; set new head pointer
    CLC
    RET                                ; set return status
40                                     ; exit
ENDP buf_putchar

;*****
;*****
45 ;*
;* BUF_INC_PTR - increment bffer pointer
;*
;* PURPOSE:
;*     This function increments the head or tail pointer associated
50 with the
;*     receive buffer.
;*
;* PARAMETERS:
;*     SI = the pointer to increment
55 ;*
;* RETURNS:
;*     SI = the next location in sngrxbuf
;*
;* REGISTERS DESTROYED:
60 ;*     SI
;*
;* GLOBALS REFERENCED:
;*     OFFSET sngrxbuf
;*     OFFSET sngrxbufend
65 ;*

```

SUBSTITUTE SHEET

```

5  ; * GLOBALS MODIFIED:
   ; *   None
   ; *
   ; * BIOS CALLS:
   ; *   None
   ; *
10  ; * DOS CALLS:
   ; *   None
   ; *
   ; * PROCEDURE CALLS:
   ; *   None
   ; *
   ; * HARDWARE ACCESS:
   ; *   None
15  ; *
   ; * NOTES:
   ; *
   ; *****
   ; *****
20
   PROC buf_inc_ptr NEAR
       INC     SI                ; increment SI
       CMP     SI,OFFSET engrxbufend ; check if the pointer has
   wrapped
       JNE     @@no_buf_wrap     ; no, continue
       MOV     SI,OFFSET engrxbuf ; yes, set back to beginning
   of buffer
   @@no_buf_wrap:
30   RET                     ; exit
   ENDP buf_inc_ptr
   ENDS
35   END
```

SUBSTITUTE SHEET

```

;*****
;*****
; Copyright (c) Absolute Software 1994, 1995
;
5  ; SNTLCOMM.ASM
;
; Contains comm routines.
;
; HISTORY:
10 ; 1995.09.05 - CCOTI
; Created.
;
;*****
;*****
15
IDEAL

%NOLIST
include "SENTINEL.INC"
20 include "SNTLCOMM.INC"
include "SNTLDATA.INC"
include "SNTLTIMR.INC"
%LIST

25 SEGMENT SNTL_SEG BYTE PUBLIC 'CODE'

;*****
;*****
30 ;
; CMFTMOUT - transmit a NAK
;
; PURPOSE:
; This functions transmits a NAK. If 3 NAK's have already been
35 transmitted,
; the transaction is terminated with an EOT.
;
; PARAMETERS:
; DX = UART Transmit Holding Register
;
40 ; RETURNS:
; Nothing
;
; NOTE:
;
45 ;*****
;*****

ASSUME CS:SNTL_SEG, DS:NOTHING, ES:NOTHING

50 PROC cmftxn timer NEAR

CMP [tx.txnackcnt], 3 ; only send 3 NAK's before
aborting
55 JE @@aborttx

MOV AL, DLNAK ; send another NAK
OUT DX, AL
INC [tx.txnackcnt]
60 MOV [tx.txnxtst], OFFSET snfgetpkt ; set state function
following tx
MOV [rx.rxtmr], TM10SEC ; set response to NAK
timeout
JMP @@exit

65 @@aborttx:
```

SUBSTITUTE SHEET

```

        MOV     AL, DLEOT                                ; send EOT to terminate
transaction
        OUT     DX, AL
        MOV     [tx.txxnxtst], OFFSET snfenrst          ; set state function
5 following tx

@@exit:
        MOV     [snfstftn], OFFSET cmftx                ; set next state function
        MOV     [tx.txxstate], OFFSET CS:cmftxcomp; set transmitter state: tx
10 complete
        MOV     [rx.rxxstate], OFFSET cmfpstx           ; reset receiver
        RETN

ENDP cmftxnak
15
        ASSUME NOTHING

;*****
;*****
;*
;* CMFTXENQ - transmit an ENQ
;*
;* PURPOSE:
;* This functions transmits a NAK. If 3 NAK's have already been
25 transmitted,
;* the transaction is terminated with an EOT.
;*
;* PARAMETERS:
;* DX = UART Transmit Holding Register
30 ;*
;* RETURNS:
;* CF = 0 if not timed out
;* CF = 1 if timed out
35 ;*
;* NOTE:
;*
;*****
;*****
40
        ASSUME CS:SNTL_SEG, DS:NOTHING, ES:NOTHING

PROC cmftxenq NEAR

45        CMP     [tx.txxenqcnt], 3                      ; only send 3 NAK's before
aborting
        JE      @@aborttx

        MOV     AL, DLENQ                                ; send another ENQ
50        OUT     DX, AL
        INC     [tx.txxenqcnt]                          ; increment transmitted ENQ
count
        MOV     [tx.txxnxtst], OFFSET snfgetpkt         ; set state function
following tx
55        MOV     [rx.rxxtmr], TM10SEC                   ; set response to ENQ
timeout
        JMP     @@exit

@@aborttx:
60        MOV     AL, DLEOT                                ; send EOT to terminate
transaction
        OUT     DX, AL
        MOV     [tx.txxnxtst], OFFSET snfenrst          ; set state function
following tx
65        MOV     [rx.rxxstate], OFFPSET cmfpstx         ; reset receiver

```

SUBSTITUTE SHEET


```

@@exit:
    MOV    [engstftn], OFFSET cmftx      ; set next state function
    MOV    [tx.txxstate], OFFSET CS:cmtxcomp; set transmitter state: tx
complete
    RETN
5
ENDP cmftxeng

10
    ASSUME NOTHING

;*****
;*****
15
;*
;* CMFPRPMDM - prepare to transmit modem string
;*
;* PURPOSE:
;*   This function prepares the transmit structure before initiating
;*   transmission of a string to the modem.
20
;*
;* PARAMETERS:
;*   BX => the string to transmit (see note below)
;*
;* RETURNS:
;*   Nothing
25
;*
;* REGISTERS DESTROYED:
;*
;* GLOBALS REFERENCED:
30
;*
;* GLOBALS MODIFIED:
;*
;* BIOS CALLS:
;*   None
35
;*
;* DOS CALLS:
;*   None
;*
;* PROCEDURE CALLS:
40
;*   None
;*
;* HARDWARE ACCESS:
;*   None
;*
;* NOTE:
45
;*   BX points to the length of the string to transmit, which is
;*   preceded in
;*   memory by the string (eg. AT<CR>3).
;*
50
;*****
;*****

    ASSUME CS:SNTL_SEG, DS:SNTL_SEG, ES:NOTHING

55
PROC cmfprpmdm

    MOV    AL, [BX]                      ; get the length of the
packet
    MOV    [BYTE LOW tx.txxpktlen], AL
    MOV    [BYTE HIGH tx.txxpktlen], 0
    SUB    BX, [tx.txxpktlen]            ; set pointer to start of
string
    MOV    [tx.txxbufp], BX
    MOV    [tx.txxpkttyp], CMTXMDHPKT    ; transmitting modem packet
65
    MOV    [tx.txxtmr], TM1SEC           ; set maximum transmit time

```

SUBSTITUTE SHEET

```

    MOV    [tx.txxtxng], 0                ; clear transmission in
progress flag

    MOV    [sngstftn], OFFSET cmftx      ; next state: transmit
5    MOV    [rx.rxxxtmr], TM6SEC         ; wait 5 seconds after tx
for rx

    RETN

10    ENDP cmfprpmdm

        ASSUME NOTHING

15    ;*****
;*****
;*
;* CMFTX - transmit state machine
;*
20    ;* PURPOSE:
;*   This function acts as the transmitter state machine performing
all packet
;*   transmissions and data-link ACK's, NAK's, and ENQ's.
;*
25    ;* PARAMETERS:
;*   None
;*
;* RETURNS:
;*   Nothing
30    ;*
;* REGISTERS DESTROYED:
;*
;* GLOBALS REFERENCED:
;*
35    ;* GLOBALS MODIFIED:
;*
;* BIOS CALLS:
;*   None
;*
40    ;* DOS CALLS:
;*   None
;*
;* PROCEDURE CALLS:
;*   None
45    ;*
;* HARDWARE ACCESS:
;*   UART (IN LSR, OUT THR)
;*
;* NOTE:
50    ;*
;*****
;*****

55    ASSUME CS:SNTL_SEG, DS:SNTL_SEG, ES:NOTHING

    PROC cmftx

        CMP    [tx.txxtmr], 0            ; has the transmitter been
on too long?
60    JE      cmtxrst                    ; yes, reset transmitter
and Sentinel

; no, continue
; ensure THR is empty.
; get DX = LSR

    MOV    DX, [sngmdmptradd]

```

SUBSTITUTE SHEET

```

      ADD    DX, LSR
      is empty
      IN     AL, DX
      it is
5      TEST  AL, 00100000b
      where TX ISR
      JZ     @@exit
      world the
10     routine is

      MOV    DX, (sngmdmprtadd)
      address
15     CMP    [tx.txxtxing], 1
      JE     cmcont
      MOV    [tx.txxdatcnt], 0
      count
20     MOV    [tx.txxtxing], 1
      progress flag
      CMP    [tx.txxpkttyp], CMTXDLNAK
      JE     cmtxnak
      CMP    [tx.txxpkttyp], CMTXDLACK
25     JE     cmtxack
      CMP    [tx.txxpkttyp], CMTXDLENQ
      JE     cmtxenq
      CMP    [tx.txxpkttyp], CMTXDLEOT
      JE     cmtxeot
30     CMP    [tx.txxpkttyp], CMTXMDMPKT
      packet?
      JNE    cmprpdata
      MOV    [tx.txxstate], OFFSET CS:cmtxdata; no, must be data packet
35     segment
      JMP    cmcont
      cmprpdata:
      MOV    [tx.txxstate], OFFSET CS:cmtxstx ; transmitting data packet
      cmcont:
40     JMP    [tx.txxstate]

      cmtxstx:
      MOV    AL, DLSTX
      OUT    DX, AL
      MOV    [tx.txxlrc], 0
45     MOV    [tx.txxstate], OFFSET CS:cmtxlenlsb ; clear LRC checksum
      JMP    @@exit

      cmtxlenlsb:
50     MOV    AL, [BYTE LOW tx.txxpktlen]
      OUT    DX, AL
      XOR    [tx.txxlrc], AL
      MOV    [tx.txxstate], OFFSET CS:cmtxlenmsb
      JMP    @@exit

55     cmtxlenmsb:
      MOV    AL, [BYTE HIGH tx.txxpktlen]
      OUT    DX, AL
      XOR    [tx.txxlrc], AL
      MOV    [tx.txxstate], OFFSET CS:cmtxtype
60     JMP    @@exit

      cmtxtype:
      MOV    AL, [tx.txxtype]
65     OUT    DX, AL
      XOR    [tx.txxlrc], AL

```

SUBSTITUTE SHEET

```

MOV     [tx.txxstate], OFFSET CS:cmtxstype
JMP     @@exit

5      cmtxstype:
      MOV     AL, [tx.txxstype]
      OUT     DX, AL
      XOR     [tx.txxlrc], AL
      MOV     [tx.txxstate], OFFSET CS:cmtxdata
10     JMP     @@exit

      cmtxdata:
      MOV     SI, [tx.txxbufp]
      ADD     SI, [tx.txxdatcnt] ; transmit the next byte
      MOV     AL, [SI]
15     OUT     DX, AL
      XOR     [tx.txxlrc], AL ; update the LRC
      INC     [tx.txxdatcnt] ; increment data byte index
      DEC     [tx.txxpktlen] ; decrement data bytes to
20     transmit
      JNZ     @@exit ; and exit if more to send

      CMP     [tx.txxpkttyp], CMTXMDMPKT ; transmission complete,
      packet? ; transmitting modem
      JNE     cmtxsetetx ; no, data packet, set to
25     finish tx
      MOV     [tx.txxstate], OFFSET CS:cmtxcomp; yes, transmission
      complete
      JMP     @@exit
30     cmtxsetetx:
      MOV     [tx.txxstate], OFFSET CS:cmtxetx ; or set next state tx data
      packet
      JMP     @@exit

35     cmtxetx:
      MOV     AL, DLETX
      OUT     DX, AL
      XOR     [tx.txxlrc], AL
      MOV     [tx.txxstate], OFFSET CS:cmtxcomp
40     JMP     @@exit

      cmtxlrc:
      MOV     AL, [tx.txxlrc]
      OUT     DX, AL
      MOV     [tx.txxstate], OFFSET CS:cmtxcomp
45     JMP     @@exit

      cmtxack:
      MOV     AL, DLACK
      OUT     DX, AL
      MOV     [tx.txxstate], OFFSET CS:cmtxcomp
50     JMP     @@exit

      cmtxnak:
      CALL    cmftxnak
55     JMP     @@exit

      cmtxeng:
      CALL    cmftxeng
      JMP     @@exit
60

      cmtxetot:
      MOV     AL, DLEOT
      OUT     DX, AL
      MOV     [tx.txxstate], OFFSET CS:cmtxcomp
65     JMP     @@exit

```

SUBSTITUTE SHEET

```

cmtxcomp:
  MOV [tx.txxtxing], 0          ; transmission complete
  progress flag                 ; clear transmission in
  MOV AX, [tx.txxnxtst]        ; move onto the next state
5  function
  MOV [sngstftn], AX
  JMP @@exit

cmtxrst:
10  MOV [tx.txxtxing], 0          ; transmitter timeout
  progress flag                 ; clear transmission in
  MOV [sngstftn], OFFSET snfsrst ; next state: reset
  Sentinel

15  @@exit:
  RET

ENDP cmftx

20  ASSUME NOTHING

;*****
;*****
25  ;*
;* CMFPACK - process expected ACK
;*
;* PURPOSE:
;*   This functions tests for an acknowledgement from the CT Server.
30  ;* PARAMETERS:
;*   None
;*
;* RETURNS:
;*   Nothing
35  ;*
;* NOTE:
;*
;*****
40  ;*****

  ASSUME CS:SNTL_SEG, DS:NOTHING, ES:NOTHING

45  PROC cmfpack NEAR

  CMP AL, DLACK                 ; ACK received?
  JNE @@testnak                 ; no, test for NAK

  MOV [rx.rxxstate], OFFSET cmfpstx ; yes, transfer complete go
  RETN                          ; await another packet

  @@testnak:
  CALL cmfpnak                  ; treat as potential NAK

55  @@exit:
  RETN

ENDP cmfpack
  ASSUME NOTHING

60  ;*****
;*****
;*
;* CMFPNAK - process NAK
65  ;*
```

SUBSTITUTE SHEET

```

; * PURPOSE:
; *   This functions tests for a negative-acknowledgement from the CT
; *   Server.
5  ; *
; * PARAMETERS:
; *   AL contains the character that may be a NAK
; *
; * RETURNS:
; *   Nothing
10 ; *
; * NOTE:
; *
; * *****
; * *****
15
    ASSUME CS:SNTL_SEG, DS:NOTHING, ES:NOTHING

PROC cmfpnak NEAR

20     CMP     AL,DLNAK                ; NAK received?
        JNE     @@exit                ; no, exit

    @@cont:
        MOV     BX, OFFSET engsernum_str    ; point to string to send
25     ; CALL     ComTransInit              ; initiate retransmission
        MOV     [engstftn], OFFSET snftxchkin

    @@exit:
        RETN
30     ENDP cmfpnak
        ASSUME NOTHING

; * *****
; * *****
35 ; *
; * CMFPSTX - process STX
; *
; * PURPOSE:
; *   This functions tests for a start-of-text character.
40 ; *
; * PARAMETERS:
; *   None
; *
; * RETURNS:
; *   Nothing
45 ; *
; * NOTE:
; *
; * *****
; * *****
50
    ASSUME CS:SNTL_SEG, DS:NOTHING, ES:NOTHING

PROC cmfpstx NEAR

55     CMP     AL, DLSTX                ; STX received?
        JE      @@cont                ; yes, continue

        CALL     cmfstrx                ; no, reset receiver
60     RETN                             ; exit

    @@cont:
        MOV     [rx.rxxlrc], 0          ; clear LRC checksum
65     MOV     [rx.rxxstate], OFFSET cmfplen1 ; set next state
```

SUBSTITUTE SHEET

```

@@exit:
    RETN

5    ENDP cmfpstx
    ASSUME NOTHING

;*****
;*****
10   ;*
;* CMFPLEN1 - process first byte of length
;*
;* PURPOSE:
;*   This functions accepts the least significant byte of the length
15   field of
;*   a packet.
;*
;* PARAMETERS:
;*   None
;*
20   ;* RETURNS:
;*   Nothing
;*
;* NOTE:
;*
25   ;*****
;*****

    ASSUME CS:SNL_SEG, DS:NOTHING, ES:NOTHING
30   PROC cmfplen1 NEAR

        MOV     [BYTE LOW rx.rxxpktlen], AL    ; store LSB of length
        XOR     [rx.rxxlrc], AL                ; update LRC
35         MOV     [rx.rxxstate], OFFSET cmfplen2 ; set next state

@@exit:
    RETN

40   ENDP cmfplen1
    ASSUME NOTHING

;*****
;*****
45   ;*
;* CMFPLEN2 - process second byte of length
;*
;* PURPOSE:
;*   This functions accepts the most significant byte of the length
50   field of
;*   a packet.
;*
;* PARAMETERS:
;*   None
55   ;*
;* RETURNS:
;*   Nothing
;*
;* NOTE:
60   ;*****
;*****

    ASSUME CS:SNL_SEG, DS:NOTHING, ES:NOTHING
65   PROC cmfplen2 NEAR

```

SUBSTITUTE SHEET

```

MOV    [BYTE HIGH rx.rxxpktlen], AL    ; store LSB of length
XOR    [rx.rxxlrc], AL                ; update LRC
5      MOV    [rx.rxxstate], OFFSET cmfptype ; set next state

@@exit:
    RETN

10      ENDP cmfplen2
        ASSUME NOTHING

;*****
;*
15      ;* CMFPTYPE - process packet type
;*
;* PURPOSE:
;*   This functions accepts the packet type field.
20      ;* PARAMETERS:
;*   None
;*
;* RETURNS:
;*   Nothing
25      ;* NOTE:
;*
;*****
30      ASSUME CS:SNTL_SEG, DS:NOTHING, ES:NOTHING

PROC cmfptype NEAR

35      MOV    [rx.rxxtype], AL          ; store packet type
XOR    [rx.rxxlrc], AL                  ; update LRC
DEC    [rx.rxxpktlen]                   ; decrement bytes remaining
MOV    [rx.rxxstate], OFFSET cmfstyp   ; set next state

40      @@exit:
        RETN

ENDP cmfptype
45      ASSUME NOTHING

;*****
;*
50      ;* CMFPSTYP - process packet subtype
;*
;* PURPOSE:
;*   This functions accepts the packet subtype field.
55      ;* PARAMETERS:
;*   None
;*
;* RETURNS:
;*   Nothing
60      ;* NOTE:
;*
;*****
65      ASSUME CS:SNTL_SEG, DS:NOTHING, ES:NOTHING

```

SUBSTITUTE SHEET


```
PROC cmfpstyp NEAR
5      MOV     [rx.rxxstype],AL           ; store packet subtype
      XOR     [rx.rxxlrc],AL             ; update LRC
      DEC     [rx.rxxpktlen]             ; decrement bytes remaining
      JNZ     @@cont                     ; continue if more data
      MOV     [rx.rxxstate], OFFSET cmfpetx ; expect ETX next if over
      JMP     @@exit
10
      @@cont:
      MOV     [rx.rxxstate], OFFSET cmfpdata ; set next state
      MOV     count, [rx.rxxbytcnt], 0      ; clear the received byte
15
      @@exit:
      RETN
20
      ENDP cmfpstyp
      ASSUME NOTHING

;*****
;*****
25      ;*
;* CMFPDATA - process packet data
;*
;* PURPOSE:
;*   This functions accepts the packet data field.
30      ;*
;* PARAMETERS:
;*   None
;*
;* RETURNS:
;*   Nothing
35      ;*
;* NOTE:
;*
;*****
;*****
40
      ASSUME CS:SNL_SEG, DS:NOTHING, ES:NOTHING
      PROC cmfpdata NEAR
45
      MOV     SI, [rx.rxxbufp]           ; get offset to store data
      ADD     SI, [rx.rxxbytcnt]
      MOV     [SI], AL                   ; store packet data
      XOR     [rx.rxxlrc], AL             ; update LRC
50      INC     [rx.rxxbytcnt]           ; increment data byte count
      DEC     [rx.rxxpktlen]             ; decrement bytes remaining
      to receive
      JNZ     @@exit                     ; and exit if more to come
      MOV     [rx.rxxstate], OFFSET cmfpetx ; or set next state if
55      finished
      @@exit:
      RETN
60
      ENDP cmfpdata
      ASSUME NOTHING

;*****
;*****
65      ;*
```

SUBSTITUTE SHEET

```

; * CMFPETX - process ETX
; *
; * PURPOSE:
; *   This functions accepts the packet ETX delimiter.
5  ; *
; * PARAMETERS:
; *   None
; *
; * RETURNS:
10 ; *   Nothing
; *
; * NOTE:
; *
; * .....
15 ; * .....

      ASSUME CS:SNL_SEG, DS:NOTHING, ES:NOTHING

PROC cmfpetx NEAR

20      XOR    [rx.rxxlrc],AL          ; update LRC
      CMP     AL,DLETX                ; test for ETX
      JE      @@cont                 ; ETX rx'd, continue

25      CALL   cmfrstrx                ; ETX not rx'd, reset rx'r
      JMP     @@exit

@@cont:
30      MOV     [rx.rxxstate], OFFSET cmfplrc ; set next state

@@exit:
      RETN

35      ENDP cmfpetx
      ASSUME NOTHING

; * .....
; *
40 ; * CMFPLRC - process LRC
; *
; * PURPOSE:
; *   This functions accepts the packet LRC checksum.
; *
45 ; * PARAMETERS:
; *   None
; *
; * RETURNS:
50 ; *   Nothing
; *
; * NOTE:
; *
; * .....
55 ; * .....

      ASSUME CS:SNL_SEG, DS:NOTHING, ES:NOTHING

PROC cmfplrc NEAR

60      IF 0
      CMP     AL, [rx.rxxlrc]          ; test for valid LRC
      JE      @@cont                 ; LRC valid, continue
      CALL    cmfrstrx                ; LRC invalid, reset rx'r
      RETN                               ; and exit
65      ELSE

```

SUBSTITUTE SHEET

```

        CMP     AL, 0
        JE      @@cont                ; test for 0 for now
                                        ; LRC valid, continue

5      @@nak:
        MOV     [engetftn], OFFSET cmftx ; LRC invalid, send a NAK
        MOV     [tx.txxpkttyp], CMTXDLNAK ; set next state: transmit
        RETN    ; set packet type: send NAK
        ENDIF ; exit

10     @@cont:
        MOV     [engetftn], OFFSET cmftx ; set next Sentinel state
        function
        MOV     [tx.txxpkttyp], CMTXDLACK ; transmitting an ACK
        MOV     [tx.txxtmr], TM1SEC      ; give tx one second to
15     complete
        MOV     [tx.txxnxtst], OFFSET cmfprdata ; set state fuction
        following tx

20     @@exit:
        RETN

        ENDP cmfplrc
        ASSUME NOTHING

25     ;*****
        ;*
        ;* CMFGETNEXT - get next call date
        ;*
30     ;* PURPOSE:
        ;* This functions extracts the next call date from a received
        ;* packet.
        ;*
35     ;* PARAMETERS:
        ;* None
        ;*
        ;* RETURNS:
        ;* Nothing
40     ;*
        ;* NOTE:
        ;*
        ;*****

45     ASSUME CS:SNTL_SEG, DS:NOTHING, ES:NOTHING

        PROC cmfgetnext NEAR

50     PUSH     DS
        POP      ES                ; get ES = DS

        MOV     DI, OFFSET next_call_date ; ES:DI points to
next_call_date
55     MOV      SI, [rx.rxxbufp]
data                ; DS:SI points to received

        CLD
        MOV     CX, 5                ; move up through pointers
data:            ; copy five bytes of BCD

60     REP      MOVSB                ;
                                ; YMMDDHHMM
                                ; copy the new date/time

        INC     [angdskwrt]          ; set the disk write flag
65     MOV      [angclst], SNCALLPASS ; set the call status

```

SUBSTITUTE SHEET

```

    MOV     AX, [sngmdmprt]                ; set default modem for
next call                                     ;
    MOV     [modem_default_port], AX      ; call based upon current
port
5
    @@exit:
    RETN

10
    ENDP cmfgetnext
    ASSUME NOTHING

    IF 0
    ;*****
    ;*
15
    ;* CMFDISABLE - disable Sentinel
    ;*
    ;* PURPOSE:
    ;*   This functions disables the Sentinel based upon a packet
20
    ;*   received from the tracking server. The Sentinel is disabled by recording a
    ;*   call date and time of 0xFFFFFFFF.
    ;*
    ;* PARAMETERS:
25
    ;*   None
    ;*
    ;* RETURNS:
    ;*   Nothing
30
    ;* NOTE:
    ;*
    ;*****
35
    ASSUME CS:SNTL_SEG, DS:NOTHING, ES:NOTHING

    PROC cmfdisable NEAR

40
    PUSH    DS
    POP     ES                            ; get ES = DS

    MOV     DI, OFFSET next_call_date    ; ES:DI points to
next call date
45
    MOV     SI, OFFSET rx.rxxdata        ; DS:SI points to received
data

    CLD
    MOV     CX, 5                        ; move up through pointers
50
    data:                                     ; copy five bytes of BCD
    REP     MOVSB                        ;
    ;*                                     ;
    ;*   YYMMDDHHMM
    ;*   ; copy the new date/time
55
    INC     [sngdskwrt]                  ; set the disk write flag

    @@exit:
    RETN

60
    ENDP cmfdisable
    ASSUME NOTHING
    ENDIF

    ;*****
    ;*
65
    ;*

```

SUBSTITUTE SHEET

```

; * CMFPRSDATA - parse received data
; *
; * PURPOSE:
; *   This functions parses received data and takes appropriate
5  action.
; *
; * PARAMETERS:
; *   None
; *
10 ; * RETURNS:
; *   Nothing
; *
; * NOTE:
15 ; *
; * .....
; * .....

      ASSUME CS:SNL_SEG, DS:NOTHING, ES:NOTHING

20  PROC cmfprdata NEAR

      MOV     AL, [rx.rxxstype]           ; test for valid data type
      CMP     AL, SNSERVER
      JNE     @@reset

25      MOV     AL, [rx.rxxstype]           ; test for valid subtype
      CMP     AL, SNNEXTCALL             ; test for next call packet
      JNE     @@nxtest1
      CALL    cmfgetnext                  ; extract next call date
30  from packet
      JMP     @@reset

      @@nxtest1:
      IF 0
35      CMP     AL, SNDISABLE             ; test for disable packet
      JNE     @@nxtest2
      CALL    snfdisable                 ; disable Sentinel
      ENDIF

40      @@reset:
      CALL    cmfstrx                     ; reset receiver

      @@exit:
      RETN

45  ENDP cmfprdata
      ASSUME NOTHING

50  ; * .....
; * .....
; * CMFRSTRX - reset the receiver
; *
; * PURPOSE:
55 ; *   This functions resets the receiver.
; *
; * PARAMETERS:
; *   None
; *
60 ; * RETURNS:
; *   Nothing
; *
; * NOTE:
; *

```

SUBSTITUTE SHEET

```
*****
*****
5      ASSUME CS:SNTL_SEG, DS:NOTHING, ES:NOTHING
      PROC cmfrstrx NEAR
          MOV     [rx.rxxstate],OFFSET cmfpstx      ; reset receiver state
10 machine
          MOV     [sngetftn],OFFSET snfsrst         ; reset the Sentinel to
          active mode
          MOV     [Sentinel_state],SNSTACTIVE
15 @@exit:
          RETN
      ENDP cmfrstrx
20      ASSUME NOTHING
      ENDS
      END
```

SUBSTITUTE SHEET

```
*****
*****
; Copyright (c) Absolute Software 1994, 1995
;
5 ; SNTLCOMV.ASM
;
; Contains the comm ISR routine.
;
; HISTORY:
10 ; 1995.09.05 - CCOTI
; Created.
;
*****
*****
15
IDEAL

$NOLIST
include "SENTINEL.INC"
20 include "SNTLCOMV.INC"
include "SNTLDATA.INC"
include "SNTLBUFF.INC"
include "SNTLAPI.INC"
$LIST
25

SEGMENT SNTL_SEG BYTE PUBLIC 'CODE'

;*****
;*****
30 ;
; CMFISR - communications interrupt service routine
;
; PURPOSE:
; This function implements the communications ISR that supports
35 bith receiving and transmitting data. This function hooks the
system's communications port interrupt (IRQ 4/3).
;
; PARAMETERS:
; None
;
; RETURNS:
; Nothing
45 ;
; GLOBALS REFERENCED:
; sngmdmprtadd
;
; GLOBALS MODIFIED:
50 ; sngincmiser - Incremented on entrance, decremented on exit.
; sngstftn - set to error handler if error is detected.
;
; BIOS CALLS:
; None
55 ;
; DOS CALLS:
; NONE
;
; PROCEDURE CALLS:
60 ; cmftxbyte, buf_putchar
;
; HARDWARE ACCESS:
; UART (IN IIR, I/O MCR, IN RDR)
;
```

SUBSTITUTE SHEET

```

*****
*****
5      ASSUME CS:SNTL_SEG, DS:NOTHING, ES:NOTHING

      PROC cmflier FAR

          PUSH    AX                ; save registers
10         PUSH    DX
          PUSH    SI
          PUSH    DS

          PUSH    CS                ; set DS
15         POP     DS
          ASSUME DS:SNTL_SEG

          INC     [engincmiser]      ; set ISR in progress flag

      IFDEF Debug
20         INC     [engcomcnt]      ; increment comm ISR count
      ENDIF

      @@check_iir:
      ; Check the reason for the call (error, ready to send, data
25      received).
          MOV     DX, [sngmdmptadd]  ; get interrupt
          ADD     DX, IIR            identification register
          IN      AL, DX
30         TEST    AL, 00000100b     ; test for receive
          JNZ     DataReceive        ; proceed with data
          JNZ     DataSend           ; proceed with data
35         TEST    AL, 00000010b     ; test for transmit
          JNZ     DataSend           ; proceed with data
          JNZ     DataSend           ; proceed with data
40         TEST    AL, 00000010b     ; test for transmit
          JNZ     DataSend           ; proceed with data
          JNZ     DataSend           ; proceed with data

      @@error:
      IFDEF Debug
          INC     [sngcomerr]
      ENDIF
45      ; Check the status of the error.
          MOV     DX, [sngmdmptadd]  ; reading the register
          ADD     DX, LSR            clears the error
          IN      AL, DX
50         JMP     @@end

      DataSend:
      ;      CALL    cmftxbyte
55         JMP     @@end

      DataReceive:
      ; First, turn off RTS.
          MOV     DX, [sngmdmptadd]
          ADD     DX, MCR
60         IN      AL, DX            ; Move DX to MCR.
          IODELAY
          AND     AL, 11111101b     ; turn off RTS
          OUT     DX, AL
          IODELAY
65      Receive:

```

SUBSTITUTE SHEET


```

IFDEF Debug
    INC      [received_count]
ENDIF
5      MOV     DX, [sngmdmptrtadd]      ; DX = RDR.
      IN      AL, DX                  ; AL = received byte.
      IODELAY
      CALL    buf_putchar              ; Put the byte into the
buffer.
10
      ; Check if there is another request pending.
      ADD     DX, 2
      IN      AL, DX                  ; Move to IIR reg.
      IODELAY
15      TEST    AL, 00000001b
      JZ      @@check_iir
      @@end:
20      MOV     AL, 20h                ; signal end of interrupt
to PIC      OUT     20h, AL
      MOV     DX, [sngmdmptrtadd]      ; get the modem control
25      ADD     DX, MCR
      IN      AL, DX
      IODELAY
30      OR      AL, 00000010b          ; turn RTS back on
register     OUT     DX, AL            ; set the modem control
      IODELAY
35      DEC     [sngincmiser]          ; clear ISR in progress
flag
      ASSUME DS:NOTHING
40      POP     DS
      POP     SI                      ; recover registers
      POP     DX
      POP     AX
      IRET
45      ENDP cmfiser
      ASSUME NOTHING
      ENDS
50      END
```

SUBSTITUTE SHEET

```
;*****
*****
; * Copyright (c) Absolute Software 1994, 1995
; *
5  ; * SNTLDATA.ASM
; *
; * Contains the global data segment for the sentinel.
; *
; * HISTORY:
10 ; *   1995.09.05 - CCOTI
; *               Created.
; *
;*****
*****
15
IDEAL

$NOLIST
include "SENTINEL.INC"
include "SNTLTIMR.INC"
include "SNTLJTBL.INC"
include "SNTLCOMM.INC"
$LIST

25 SEGMENT SNTL_SEG BYTE PUBLIC 'CODE'

; Transient variables
*****

30          sngatftn          DW NEAR PTR OFFSET ActiveRoutine    ; CCOTI
          Sentinel_state     DB SNSTACTIVE

; Scratch vars to store the current port info being used.
35          sngmdmprt         DW ?
          sngmdmprtint        DW ?
          sngmdmprtadd        DW ?

; Previous ISR vectors.
40          sngprvtmr         DD FAR PTR 0
          sngprvcom           DD FAR PTR 0
          sngprvdskl          DD FAR PTR 0
          sngprvint2f         DD FAR PTR 0

; ROR'd to limit updating the real-time clock once every 16 ticks (see
45 ActiveRoutine).
          cycle_var           DW 0001h

          win_flag            DB 0
          win_vm              DB 1
50
          sngincmisr          DB 0

          send_buf_len        DW 0
          send_buf_ptr        DW BYTE PTR 0
55
          count               sngcomcnt          DW 0          ; comm. interrupt

          sngcomerr           DB 0          ; comm. error count
          TimerISR_count      DW 0          ; timer interrupt count
60          sent_count         DW 0          ; bytes transmitted
          received_count      DW 0          ; byte received
          sngflcnt            DB 0
          sngclst              DB SNCALLNA
          sngcomhk            DB 0
65          sngsuspend         DB 0
```

SUBSTITUTE SHEET

```

    sngdlytmr          DW 0
    sngint2ftmr        DW 0
manager              ; wait 2 minutes for an XMS
    sngprtdlytmr      DW 0
    sngdeflect        DB 1
deflection flag      ; Sentinel disk
    dkgcyl            DW ?
    dkgsctr            DB ?
    sngapifl          DB 0
    sngpwd1            DW 'FO'
password1            ; API failed request count
    sngpwd2            DW 'AD'
password2            ; API request user

15 ; Port info..
    modem_default_port DW 0
    port_table         DW 03F8h, 000Ch, \
                        02F8h, 000Bh, \
                        0000h, 0000h, \
                        02E8h, 000Dh
20
    PORT_TABLE_SIZE = 4

25 ; Disk location of data sector.
    data_cyl_sect      DW 0
    data_head_drive    DW 0
    sngdskwrt          DB 0

30 ; Output strings.
    init_str_num        DW 0
    init_str_table      DW 5 DUP( 0 )
35
    INIT_STR_TABLE_SIZE = 6
    dial_str_num        DW 0
    dial_str_table      DW 4 DUP( 0 )
    DIAL_STR_TABLE_SIZE = 5
40
    dial_number_start   DB "18003396122", 0Dh
    LABEL dial_number   BYTE
    dial_number_len     DB 12

45
    LABEL sn_packet_start UNKNOWN
    stx_byte            DB 02h
    lsb_length_byte     DB ?
    msb_length_byte     DB ?
50
    LABEL sn_text_start UNKNOWN
    text_type           DB 0
    text_sub_type       DB 0
    LABEL sn_data_start UNKNOWN
    sngsernum           DB 6 DUP( 0 )
55
    LABEL now_date       UNKNOWN
    now_year            DB 1
    now_month           DB 1
    now_day             DB 1
    now_hour            DB 1
    now_minute          DB 1
60
    LABEL sn_data_end    UNKNOWN
    etx_byte            DB 03h
    lrc_byte            DB ?
    LABEL sn_packet_end  UNKNOWN
    LABEL sngsernum_str  UNKNOWN
65
    sngsernum_str_len   DB sn_packet_end - sn_packet_start

```

SUBSTITUTE SHEET

```

        sngdatalen          DB sn_data_end - sn_data_start
;END MOD

        ; initialize receive
5      structure
        rx                  RXZCM    < OFFSET cmpack, \
                                0, 0, 0, 0, 0, 0, \
                                OFFSET CS:nextcall_text >

10     ; initialize transmit
        structure
        tx                  TXZCM    < 0, 0, 0, 0, 0, 0, 0, 0, \
                                0, 0, 0, 0, \
                                OFFSET CS:sngtxbuf >

15     ; Result tables.
        command_result_table_len DB 3
        command_result_table     DW 3 DUP( 0 )

        mdm_init_result_table_len DB 2
20     mdm_init_result_table     DW 2 DUP( 0 )

        dial_result_table_len DB 6
        dial_result_table       DW 6 DUP( 0 )

25     connect_result_table_len DB 4
        connect_result_table     DW 10 DUP( 0 )

        ; Modem and result string pool.
        string_pool              DB 127 DUP( 0 )

30     modem_find_str_start      DB 'ATZ', 0Dh
        LABEL modem_find_str     UNKNOWN
        modem_find_str_len       DB 4

35     ; next call date
        LABEL next_call_date     UNKNOWN
        next_call_year           DB OFFh
        next_call_month          DB OFFh
        next_call_day            DB OFFh
40     next_call_hour            DB OFFh
        next_call_minute         DB OFFh

        sngrxbufhd              DW 0          ; receive buffer
        sngrxbufctl              DW 0
45     LABEL sngrxbufst          UNKNOWN
        sngrxbuf                 DB 80h DUP( 0 )
        LABEL sngrxbufend        UNKNOWN

        nextcall_text            DB 05h DUP( 0 )

50     sngtxindex                DB 0          ; transmit buffer
        LABEL sngtxbufst        UNKNOWN
        sngtxbuf                 DB 7Bh DUP( 0 )
        LABEL sngtxbufend        UNKNOWN

55     ; Result jump tables.

        ; Table for ModemFind
60     find_jump_table           DW NEAR PTR find_timeout ; TIMEOUT
        DW NEAR PTR find_ok      ; NO
        CARRIER (NOTE 1)

        DW NEAR PTR find_timeout ; ERROR
65     DW NEAR PTR find_ok        ; OK

```

SUBSTITUTE SHEET

```

; NOTE 1: 29 March 1995 - DBOYD
;          USR modem (and maybe others) does not return <NO
CARRIER>
;          when the server disconnects. <NO CARRIER> returned
5  when next
;          signal (command or control line) sent to modem.
;          Sometimes this
;          response is sent before the next command, sometimes
10 after. When
;          the Sentinel receives this response to a modem query
;          (<AT>) it
;          should interpret it as <OK>.

; Table for ModemInit.
15 init_jump_table
      DW NEAR PTR init_error      ; TIMEOUT
      DW NEAR PTR init_error      ; ERROR
      DW NEAR PTR init_ok         ; OK

; Table for dial results.
20 dial_jump_table
      DW NEAR PTR dial_error      ; TIMEOUT
      DW NEAR PTR dial_error      ; ERROR
      DW NEAR PTR dial_busy       ; BUSY
      DW NEAR PTR dial_no_tone    ; NO DIAL
25 TONE
      DW NEAR PTR dial_no_carr    ; NO
CARRIER
      DW NEAR PTR dial_server     ; Server
30 Query (NAK)
      DW NEAR PTR dial_server     ; Server
Query (ENQ)

      cnct_jump_table
      DW NEAR PTR cnct_error      ; TIMEOUT
35 CARRIER
      DW NEAR PTR cnct_error      ; NO
EOT
      DW NEAR PTR cnct_eot        ; Server
ENQ
      DW NEAR PTR cnct_enq        ; Server
40 NAK
      DW NEAR PTR cnct_nak        ; Server
ACK
      DW NEAR PTR cnct_ack        ; Server
ENDS
45 include "SNTLDATA.INC"
      END
```

SUBSTITUTE SHEET

```

;*****
;*****
; * Copyright (c) Absolute Software 1994, 1995
5 ; *
; * SNTLI13V.ASM
; *
; * Contains INT 13 ISRs and disk deflection routines.
; *
10 ; * HISTORY:
; *   1995.09.05 - CCOTI
; *               Created.
; *
;*****
;*****
15
IDEAL

%NOLIST
include "SENTINEL.INC"
include "SNTLI13V.INC"
20 include "SNTLDATA.INC"
include "SNTLI2FV.INC"
include "SNTLTIMR.INC"
include "SNTLAPI.INC"
25 %LIST

SEGMENT SNTL_SEG BYTE PUBLIC 'CODE'

;*****
;*****
;
; DKFDPLRD - Disk deflect read
;
; PURPOSE:
35 ;   This function performs disk read deflections by filling up the
destination
;   buffer with erroneous characters.
;
; PROTOTYPE:
40 ;
; PARAMETERS:
;   AL = number of sectors to read (must be non-zero)
;   CH = low eight bits of cylinder number
;   CL = sector number 1-63 (bits 0-5)
45 ;   high two bits of cylinder number (bits 6-7, hard disk
only)
;   DH = head number
;   DL = drive number (bit 7 set for hard disk)
;   ES:BX => data destination
50 ;
; RETURNS:
;   The flags register as set by the ROM interrupt 13 handler:
;   - CF = 0 if successful
;   AH = status
55 ;   AL = number of sectors transferred
;
; NOTE:
;
;*****
;*****
60

        ASSUME CS:SNTL_SEG, DS:NOTHING, ES:NOTHING

65 PROC dkfdplrd NEAR

```

SUBSTITUTE SHEET

```

        MOV     DI, BX                ; get offset of destination
buffer  PUSH     AX                    ; store disk access
5  parameters  PUSH     DS            ; store register
        PUSH     CS                    ; set DS:SI pointer
        POP      DS
        MOV     SI, OFFSET fillr

10  @@dflloop:  CLD                    ; deflect loop
        MOV     CX, 100h              ; set pointers to increment
        = 1 sector)                  ; fill 256 words (512 bytes)

15  @@dflect:  MOVSW                   ; single sector deflection
destination  ; copy filler to
        DEC     SI                    ; decrement source pointer
20  DEC     SI                        ; by 2 for word moves
        LOOP    @@dflect

        DEC     AL                    ; decrement the number of
sectors to fill
25  JNZ      @@dflloop

        POP     DS                    ; restore register
        POP     AX                    ; restore disk access
parameters  MOV     AH, 0             ; set success parameters
30  and exit  CLC
        RET

35  fillr:    FILL EQU 0f6f6h

ENDP dkfdflrd
        ASSUME NOTHING

40  ;*****
;*****
;
45  ; DKFDFLMBR - Disk deflect MBR access
;
; PURPOSE:
; This function performs disk deflection on attempted access to
; MBR sector.
; Access is deflected from our subloader in the MBR to the
50  original MBR.
;
; PROTOTYPE:
;
; PARAMETERS:
55  ; AH = disk function: 0x02 = disk read
;                               0x03 = disk write
; AL = number of sectors to read (must be non-zero)
; CH = low eight bits of cylinder number
60  ; CL = sector number 1-63 (bits 0-5)
;       high two bits of cylinder number (bits 6-7, hard disk
only)
; DH = head number
; DL = drive number (bit 7 set for hard disk)
65  ; ES:BX => the data source (writes) or data destination (reads)
;

```

SUBSTITUTE SHEET

```

; RETURNS:
;   The flags register as set by the ROM interrupt 13 handler:
;   - CF = 0 if successful
;   AH = status
5  ;   AL = number of sectors transferred
;
; NOTE:
;
10 *****
;
; ASSUME CS:SNTL_SEG, DS:NOTHING, ES:NOTHING
;
15 PROC dkfdflmbr NEAR
;
;   CMP     AH, 02h           ; read access to MBR?
;   JE      @@dflmbrdr       ; yes, deflect read
;   CMP     AH, 03h           ; write access to cylinder
20 0?   JE      @@dflmbrwrt    ; yes, deflect write
;
; @@dflmbrdr:
;   PUSH    CX               ; save disk access
;   parameters
25   PUSH    AX
;   MOV     CX, 0002h         ; load CX to access
;   deflected MBR
;   MOV     AL, 1            ; load AL to access a
30   single sector
;   PUSHF                    ; push flags because IRET
;   CALL    [DWORD CS:sngprvdskl] ; from original handler
;   pops flags
;   JNC     @@dflrdcnt        ; error?, no, continue
;   POP     CX               ; yes, recover access
35   parameters
;   POP     CX
;   JMP     @@exit            ; discard original AX
;                               ; exit
;
; @@dflrdcnt:
;   POP     AX               ; recover disk access
40   parameters
;   POP     CX
;   MOV     AH, 0            ; set success indication
;   CMP     AL, 1            ; all sectors read?
45   JE      @@exit          ; yes, exit
;                               ; no, load crap to the next
;   10 sectors
;   PUSH    AX               ; save disk access
;   parameters
50   MOV     AX, ES
;   buffer by
;   ADD     AX, 200h         ; by 512 (512 bytes = 1
;   sector)
;   MOV     ES, AX
55   POP     AX
;   parameters
;   DEC     AL               ; fill one less sector than
;   required
60   CALL    dkfdflrd         ; fill destination buffer
;   with junk
;
;   PUSH    AX
;   MOV     AX, ES           ; reset destination buffer
65   pointer

```

SUBSTITUTE SHEET


```

SUB      AX, 200h
MOV      ES, AX
POP      AX
INC      AL
5      sectors read      ; increment number of
MOV      AH, 0           ; set success indication
CLC
JMP      @@exit          ; exit

10      @@dfldrwr:
        PUSH      CX      ; save disk access
parameters
        PUSH      AX
15      MOV      CX, 0002h ; load CX to access
        MOV      deflected MBR
        MOV      AL, 1    ; load AL to access a
single sector
        PUSHF
20      CALL     [DWORD CS:sngprvdk1] ; push flags because IRET
        JNC      @@dfldrwr ; from original handler
        JNC      @@dfldrwr ; error? no, continue
        POP      CX      ; yes, recover access
parameters
        POP      CX
25      JMP      @@exit    ; discard original AX
        JMP      @@exit    ; exit

        @@dfldrwr:
        MOV      AH, 2    ; read in the (possibly)
30      MOV      CX, 0002h ; image of true MBR
        MOV      AL, 1
        PUSHF
        CALL     [DWORD CS:sngprvdk1] ; push flags because IRET
        JNC      @@dfldrwr ; from original handler
35      JC       @@exit    ; error? yes, exit

        table
        ; get copy of partition
40      PUSH     DS
        PUSH     ES      ; save register
        parameter ; save disk access
        PUSH     ES
        POP      DS      ; set DS
45      PUSH     CS
        POP      ES      ; set ES
        MOV      AX, BX
        ADD      AX, 0FCh
        MOV      SI, AX
50      MOV      DI, OFFSET sngprvdk1 ; get a pointer to a buffer
        MOV      CX, 100h ; prepare to move 256 bytes
        REP      MOVSB ; do the move

55      POP      ES      ; get copy of subloader
        parameter ; restore disk access
        MOV      AH, 2
        MBR      ; read subloader from the
        MOV      CX, 0001h
60      MOV      AL, 1
        PUSHF
        CALL     [DWORD CS:sngprvdk1] ; push flags because IRET
        JNC      @@exit2 ; from original handler
        JC       @@exit2 ; error? yes, exit

```

SUBSTITUTE SHEET

```

subloader                                     ; copy partition table into
    PUSH    CS                               ; set DS
    POP     DS
5    MOV     SI, OFFSET sngrxbuf             ; DS:SI => partition table
in subloader
    MOV     AX, BX
    ADD     AX, 0FCh
    MOV     DI, AX                           ; ES:DI => partition table
10   in MBR
    MOV     CX, 100h                         ; prepare to move 256 bytes
    REP     MOVSB                           ; do the move

    MOV     AH, 3                           ; write the subloader
15   MOV     CX, 0001h
    MOV     AL, 1
    PUSHF                                     ; push flags because IRET
    CALL    [DWORD CS:sngprvdskl]           ; from original handler
20   pops flags
    JC      @@exit2                         ; error? yes, exit
    MOV     AH, 2                           ; read new MBR back into
    ES:BX
25   MOV     AL, 1
    PUSHF                                     ; push flags because IRET
    CALL    [DWORD CS:sngprvdskl]           ; from original handler
    pops flags
    JC      @@exit2                         ; error? yes, exit
30   POP     DS                             ; recover register
    POP     AX                             ; recover disk access
parameters
    POP     CX
35   MOV     AH, 0                           ; set success indication
    CLC
    JMP     @@exit

@@exit2:                                     ; if exiting due to disk
40   write
    POP     DS                             ; deflection error

@@exit:
    RET

45   ENDP dkfdflmbr
      ASSUME NOTHING

50   ;*****
      ;
      ; INT13ISR - Sentinel interrupt 13 ISR
      ;
      ; PURPOSE:
55   ; This function provides the Sentinel's interrupt 13 hook for
      ; disk access.
      ; It serves two purposes: to store next-call information to disk
      ; after a
      ; transaction with the Sentinel server, and to prevent disk reads
60   ; of the
      ; sectors that contain the Sentinel.
      ;
      ; After a tracking transaction with the server, the Sentinel will
      ; have

```

SUBSTITUTE SHEET

```

; received a next-call-date that must be recorded to disk. The
Sentinel
; disk access is piggy-backed onto a disk read or write to the
5 disk that
; the Sentinel is installed on.
;
; If a program (such as a Norton Disk Editor or Anit-Virus)
attempts to
; read a section of the hard disk that the Sentinel occupies,
10 this function
; will deflect the read to the original code that occupied the
Sentinel's
; disk space.
;
15 ; Disk access other than read/writes is passed through to the
original
; interrupt 13h handler.
;
; PROTOTYPE:
;
20 ; PARAMETERS:
; AH = disk function: 0x02 = disk read
;                      0x03 = disk write
; AL = number of sectors to read (must be non-zero)
25 ; CH = low eight bits of cylinder number
; CL = sector number 1-63 (bits 0-5)
;     high two bits of cylinder number (bits 6-7, hard disk
only)
; DH = head number
30 ; DL = drive number (bit 7 set for hard disk)
; ES:BX => the data source (writes) or data destination (reads)
;
; RETURNS:
; The flags register as set by the ROM interrupt 13 handler:
35 ; - CF = 0 if successful
; AH = status
; AL = number of sectors transferred
;
; NOTE:
40 ;
;.....
;.....
45
ASSUME CS:SNTL_SEG, DS:NOTHING, ES:NOTHING

OFFSET TO PREHANDLER = PreInt13Handler - JMP_REL_OFFSET
OFFSET TO FULLHANDLER = FullInt13Handler - JMP_REL_OFFSET

50 load_time      DW ?
loaded.          ; the time the system

PROC Int13ISR FAR
JMP_SHORT_REL_OPCODE DB 0EBh
Int13 RelOffset
55 JMP_REL_OFFSET:   DB OFFSET TO PREHANDLER

PreInt13Handler:
PUSH AX
60 PUSH ES
PUSH DS
PUSH CS
POP DS
ASSUME DS:SNTL_SEG
; Check for an XMS manager.
65 MOV AX, 4300h
```

SUBSTITUTE SHEET

```

                INT      2Fh
                CMP      AL, 80h
                JE       @@XMS_Detected          ; XMS loaded, re-hook INT
5  2Fh.
    ; Check for timeout.
    MOV        AX, 0040h
    MOV        ES, AX                          ; ES = bios segment.
    MOV        AX, [ES:006Ch]                  ; Load current bios time.
    SUB        AX, [load_time]                 ; Find delta since
10  sentinel.
    CMP        AX, PREINT13_TIMEOUT            ; Check for timeout.
    JMP        @@jmp_to_full_isr
    JB         @@jmp_to_full_isr               ; If timeout, continue and
15  hook sentinel.

@@XMS_Detected:
    PUSH       BX
    PUSH       CX
    PUSH       DI
20  @@hook2F:
    MOV        BX, 002Fh
    MOV        DI, OFFSET sngprvint2f
    MOV        CX, OFFSET snfint2f
    CALL       SwapInt
25  @@hook1C:
    MOV        BX, 001Ch
    MOV        DI, OFFSET sngprvtmr
    MOV        CX, OFFSET tmfiser
    CALL       SwapInt
30  ; Enable full int13 handler.
    MOV        [Int13_RelOffset], OFFSET_TO_FULLHANDLER
    POP        DI
    POP        CX
35  POP        BX

@@jmp_to_full_isr:
    ASSUME DS:NOTHING
    POP        DS
    POP        ES
    POP        AX
40  ; JMP        [DWORD CS:sngprvdekl]          ; pass control to original
    handler

45  FullInt13Handler:
    IF TWODSKHKS
        CMP     [CS:sngdskskip], 0            ; this invocation directed
50  to skip test?
        JNE     @@passthru                    ; yes, pass control through
        to first disk hook
        MOV     [CS:sngdskskip], 1            ; set flag for (possible)
        second hook
    ENDIF
55  @@dsktst1:
    OR         AL, AL                          ; is the sector quantity
    zero?
    JNZ        @@dsktst2                      ; no, continue
60  JMP        @@passthru                    ; pass control through

    @@dsktst2:
        CMP     [CS:sngdeflect], 1            ; disk deflection enabled?
        JNE     @@piggyback                  ; no, check for piggy-back
65  access

```

SUBSTITUTE SHEET

```

5  @@dsktst3:
    CMP     DX, 0080h           ; access to Sentinel head
    and drive?
    JNE     @@piggyback        ; no, check for piggy-back
    access

10  @@dsktst4:
    CMP     CX, 000Ch           ; access to first 12
    sectors?
    Sentinel location)
    JA      @@piggyback        ; (MBR subloader and
    access                      ; no, check for piggy-back

15  PUSH     BX                 ; save important register
    MOV     [BYTE LOW CS:dkgcyl], CH ; get the cylinder
    MOV     BL, CL
    SHR     BL, 6
    AND     BL, 03h
20  MOV     [BYTE HIGH CS:dkgcyl], BL
    MOV     [CS:dkgsctr], CL    ; get the sector
    AND     [CS:dkgsctr], 3fh
    POP     BX                 ; recover important
    register

25  @@deflect:
    determined                  ; at this point it has been
    attempting to               ; that the system is
30  sectors of                  ; access the first 12
    deflect                     ; cylinder 0 and we must

35  CMP     [dkgsctr], 1        ; access starting on MBR?
    JE      @@dfmlbr           ; yes, go deflect
    read/write
    CMP     AH, 02h            ; read access to cylinder
    O?
40  JE      @@dfldr            ; yes, deflect read
    CMP     AH, 03h            ; write access to cylinder
    O?
    JE      @@dflwrt           ; yes, deflect write
    JMP     @@passthru         ; pass control through

45  @@dfmlbr:
    sector                      ; deflect access from MBR
    CALL    dkdfmlbr           ; to original MBR
    JMP     @@return2          ; exit

50  @@dfldr:
    CALL    dkdfldr            ; deflect a read
    JMP     @@return2          ; exit

55  @@dflwrt:
    MOV     AH, 0              ; deflect a write
    and exit                    ; set success parameters
    CLC
    JMP     @@return2

60  @@piggyback:
    CMP     [CS:snkgdskwrt], 1 ; does the Sentinel need
    disk access?
    JE      @@contpb           ; yes, continue piggy-back
65  JMP     @@passthru         ; pass control through

```

SUBSTITUTE SHEET

```

@@contpb:
disk                                     ; write next-call-date to
5 ready to                             ; at this point we are
to the                                ; piggy-back onto a write
10 Sentinel is on                      ; same drive that the
IF TWODSKHKS
    CMP    [sng2dskhks], 1             ; are we hooked twiced?
    JE     @@dskacc2                   ; yes, execute second
15 handler
    @@dskacc1:
    handler                             ; execute first disk
        PUSHF
        CALL [DWORD CS:sngprvdsk1]     ; push flags because IRET
20 pops flags                           ; from original handler
        JC     @@return                ; exit if disk access error
        JMP    @@contpb2
    @@dskacc2:
        PUSHF
        CALL [DWORD CS:sngprvdsk2]     ; execute second handler
25 pops flags                           ; push flags because IRET
        JC     @@return                ; from original handler
30 ELSE
        PUSHF
        CALL [DWORD CS:sngprvdsk1]     ; push flags because IRET
        JC     @@return                ; from original handler
35 ENDIF
        JC     @@return                ; exit if disk access error
    @@contpb2:
        PUSHA
        PUSH    DS
        PUSH    ES
40    PUSH    CS
        POP     DS                     ; set DS
        PUSH    CS
        POP     ES                     ; set ES
45    ASSUME    DS:SNTL_SEG
        MOV     [sngdskwrt], 0         ; clear the Sentinel flag
call.                                     ; Load registers for int13
50    MOV     AX, 0301h                ; 03=disk write; 01=1
sector
    MOV     CX, [data_cyl_sect]        ; set cylinder and sector
to write
    MOV     DX, [data_head_drive]     ; set the head and drive
55    MOV     BX, DATA_SECTOR_OFFSET
        PUSHF
        CALL [sngprvdsk1]             ; push flags because IRET
        JC     @@write_error           ; from original handler
60    JC     @@write_error             ; disk access error, jmp
here for now
    JMP     @@cleanup                 ; disk write successful
    @@write_error:
    @@cleanup:
65    ASSUME NOTHING

```

SUBSTITUTE SHEET

```

                POP      ES
                POP      DS
                POPA
5      @@return:
        IF TWODSKHKS
            MOV         [CS:sngdskskip], 0          ; clear disk access skip
            flag
        ENDIF
10     RET          2                                ; discard FLAGS from stack
        and return

        @@return2:
            ASSUME NOTHING
15     IF TWODSKHKS
            MOV         [CS:sngdskskip], 0          ; clear disk access skip
            flag
        ENDIF
20     RET          2                                ; discard FLAGS from stack
        and return

        IF TWODSKHKS
25     @@passthru:
            time
            ASSUME CS:SNTL_SEG
            CMP         [CS:sng2dskhks], 0          ; is disk access hooked
            twice?
            JNE         @@sechandle                  ; yes, pass control to
30     second hook
            JMP         [DWORD CS:sngprvdskl]        ; no, pass control to
            original handler
            @@sechandle:
35     PUSHF
            CALL        [DWORD CS:sngprvdskl2]
            JMP         @@cleanup

            @@passthru1:
            handling access                          ; earlier disk hook
40     JMP         [DWORD CS:sngprvdskl]
            handler
            ; pass control to original
            ; and it will IRET
        ELSE
            @@passthru:
45     time
            JMP         [DWORD CS:sngprvdskl]        ; pass control to original
            handler
        ENDIF
50     ENDP Int13ISR

        ASSUME NOTHING
55     ENDS

        END
```

SUBSTITUTE SHEET

```

;*****
;*****
; Copyright (c) Absolute Software 1994, 1995
;
5  ; SNTLI2FV.ASM
;
; PURPOSE:
;   Contains INT 2F ISRs used by the sentinel.
;
10 ; HISTORY:
;   1995.09.05 - CCOTI
;   Created.
;
;*****
15 ;*****

IDEAL

%NOLIST
20 include "SENTINEL.INC"
include "SNTLI2FV.INC"
include "SNTLDATA.INC"
include "SNTLTIMR.INC"
25 include "SNTLAPI.INC"
%LIST

SEGMENT SNTL_SEG BYTE PUBLIC 'CODE'

;*****
; Unmovable code.
;*****

;*****
;Routine: Int2FVect
35 ;
;Descriptor: Provides an API and RPL 2F/4A06 support
;
;*****

40     ASSUME CS:SNTL_SEG, DS:NOTHING, ES:NOTHING

PROC Int2FVect FAR
    JMP     SHORT @@entry
    NOP
45     rpl_sig    DB 'RPL'

@@entry:
    CMP     AX,4A06h
    JNE     @@next_check
50     MOV     DX,CS

@@next_check:
                                ; must be a Sentinel check
    IF 0
55     CMP     AX,SNTL_SIG1      ; proper signature provided?
    JNE     @@exit              ; no, exit
    CMP     DX,SNTL_SIG2      ; proper signature provided?
    JNE     @@exit              ; no, exit
    MOV     AX,OFFSET CS:SntlAPI ; yes, return API address
    MOV     DX,CS
60     ENDIF

@@exit:
    IRET
65     ENDP Int2FVect
```

SUBSTITUTE SHEET

ASSUME NOTHING

```

5  ;*****
;
; * SNFINT2F - interrupt 2F hook
;
; * PURPOSE:
10 ; * This is the interrupt 2F hook that supports the Sentinel API
request and
; * monitors WINDOWS activation/deactivation
;
; * PARAMETERS:
15 ; * None
;
; * RETURNS:
; * Nothing
;
; * REGSITERS DESTROYED:
20 ;
; * GLOBALS REFERENCED:
;
; * GLOBALS MODIFIED:
;
25 ; * BIOS CALLS:
; * None
;
; * DOS CALLS:
30 ; * None
;
; * PROCEDURE CALLS:
; * None
;
; * HARDWARE ACCESS:
35 ; * None
;
;*****
*****
40 ASSUME CS:SNTL_SEG, DS:NOTHING, ES:NOTHING

PROC snfint2f FAR

45 starting to load      CMP      AX, 1605h          ; check if WINDOWS is
JNE      @@check1
MOV      [BYTE CS:sngsuspend], 1      ; suspend Sentinel
until WINDOWS is loaded
50 IRET from call      PUSHF          ; push flags because
CALL     [DWORD CS:sngprvint2f]      ; to previous handler
pops flags      IRET          ; return from
interrupt
55 @@check1:
has finished loading    CMP      AX, 1608h          ; check if WINDOWS
JNE      @@check2
60 resume      MOV      [BYTE CS:sngsuspend], 0      ; allow Sentinel to
MOV      [BYTE CS:sngdlytmr], 90      ; set the delay timer
;
reset after delay
65 MOV      [WORD CS:sngstftn], OFFSET snfsrst
```

SUBSTITUTE SHEET

```

                    PUSHF
IRET from call      ; push flags because
                    CALL    (DWORD CS:sngprvint2f) ; to previous handler
5  pops flags
                    MOV     [BYTE CS:win_flag], 1 ; set WINDOWS status
flag
                    MOV     [sngincmiar], 0 ; clear
communications ISR flag
10  IRET
interrupt          ; return from

@@check2:
                    CMP     AX, 1606h ; check if WINDOWS
15  has exited
                    JNE     @@check3
                    MOV     [BYTE CS:sngsuspend], 0 ; allow Sentinel to
resume
                    MOV     [BYTE CS:sngdlytmr], 90 ; set the delay timer
20  reset after delay
                    MOV     [WORD CS:sngstftn], OFFSET snfsnrst
                    PUSHF
IRET from call      ; push flags because
25  pops flags
                    CALL    (DWORD CS:sngprvint2f) ; to previous handler
status flag
                    MOV     [BYTE CS:win_flag], 0 ; clear WINDOWS
communications ISR flag
30  MOV     [sngincmiar], 0 ; clear
IRET
interrupt          ; return from

@@check3:
35  starting exit
                    CMP     AX, 1609h ; check if WINDOWS is
                    JNE     @@check4
                    MOV     [BYTE CS:sngsuspend], 1 ; suspend Sentinel
until WINDOWS has exited
40  PUSHF
IRET from call      ; push flags because
                    CALL    (DWORD CS:sngprvint2f) ; to previous handler
pops flags
                    IRET
45  interrupt
                    ; return from

@@check4:
                    CMP     AX, 1607h ; check if WINDOWS is
50  testing for 32
                    JNE     @@check5
support
                    CMP     BX, 0010h
                    JNE     @@check5
                    CMP     CX, 0003h
                    JNE     @@check5
55  MOV     CX, 0000h ; set return value to
indicate 32-bit support
                    IRET
interrupt          ; return from

60  @@check5:
                    CMP     AX, SNTL_SIG1 ; check for API request
in AX:DX
                    ; check for signature
                    JNE     @@org
previous handler    ; no match, go to
65  CMP     DX, SNTL_SIG2

```

SUBSTITUTE SHEET

```

JNE      @@org      ; no match, go to
previous handler

5  AX:DX match, but no access yet
    CMP    (sngapif1), 10      ; more than ten
    failed API requests?
    JAE    @@apifail      ; yes, jump to
    original handler

10  in BX:CX      CMP    BX, [sngpwd1]      ; check for passwords
    JNE    @@bkdr      ; no match, check for
    backdoor password
    CMP    CX, [sngpwd2]      ; check for passwords
15  in BX:CX      JNE    @@bkdr      ; no match, check for
    backdoor password
    JMP    @@apipass      ; ok!

20  @@bkdr:      CMP    BX, [WORD sngsernum] ; check for backdoor access
    JNE    @@apifail      ; no match, increment
    failure count
25  CMP    CX, [WORD sngsernum + 2]
    JNE    @@apifail      ; no match, increment
    failure count

    @@apipass:
30  password match  MOV    AX, OFFSET CS:SntlAPI      ; signature and
    point          MOV    DX, CS      ; return API entry
    interrupt      IRET      ; return from

35  @@apifail:
    presence but failed access
    MOV    DX, 0FOADh      ; alert CTM to
40  INC    [sngapif1]
    IRET
    interrupt      ; return from

    @@org:
45  JMP    [DWORD CS:sngprvint2f]      ; pass control to
    previous handler

    ENDP snfint2f
    ASSUME NOTHING

50  ENDS

    END

```

SUBSTITUTE SHEET

```

*****
; Copyright (c) Absolute Software 1994, 1995
;
5  ; SNTLINIT.ASM
;
; contains all initialization code that is discarded from memory.
;
; HISTORY:
10 ; 1995.09.05 - CCOTI
; File created from the old SNTLINIT.ASM.
;
*****
15
SEGMENT SNTL_INIT_SEG PARA PUBLIC 'CODE'
ENDS

IDEAL

20
$NOLIST
include "SENTINEL.INC"
include "SNTLDATA.INC"
include "SNTLI2FV.INC"
25 include "SNTLI13V.INC"
$LIST

;*****
;*****
30 ;
; * SNTL_INIT_SEG - Transient segment.
; *
;*****
;*****
35 SEGMENT SNTL_INIT_SEG PARA PUBLIC 'CODE'
ASSUME CS:SNTL_INIT_SEG, DS:NOTHING

;*****
;*****
40 ; Sntlinit Header

part_sector DB 512 DUP( 0 )
boot_sector DB 512 DUP( 0 )
45 io_sector DB 512 DUP( 0 )

SntlSignature DW SNTL_SIG1, SNTL_SIG2
JMP NEAR SntlInit

50 fddataseq:
parameters ; sentinel source image
fdgssihddrv DW 0000h ; determined with Norton
Disk Editor
fdgssicylsec DW 0101h ; determined with Norton
55 Disk Editor
fdgssisec DB 11 ; written by CTM

parameters ; sentinel target image
60 fdgstihddrv DW 0000h ; written by CTM
fdgsticylsec DW 0000h ; written by CTM
fdgstisec DB 00 ; written by CTM

parameters ; subloader source image

```

SUBSTITUTE SHEET

```

    fdgslsihddrv      DW 0100h      ; determined with Norton
Disk Editor
    fdgslsicylsec     DW 0010h      ; determined with Norton
Disk Editor
    fdgslsisecc       DB 1          ; written by CTM
5
    parameters        ; subloader target image
    fdgsltihddrv      DW 0000h      ; written by CTM
10    fdgslticylsec     DW 0000h      ; written by CTM
    fdgsltiisec       DB 0          ; written by CTM
    fdginstall        DB 0          ; written by CTM to
15 activate HDD
    program            ; infection by FDD boot
    fdgdkerr          DB 0          ; disk access error count
    fdghddbshd        DW ?          ; HDD Boot Sector
20 head and drive
    fdghddbsec        DW ?          ; HDD Boot Sector cylinder
    and sector
    fdghddid          DD ?          ; HDD volume ID written by
25 CTM to prevent
    infecting wrong disk            ; FDD boot program from
30 ;*****
    ; Sntlinit entry points (stack= AX,BX,CX,DX,DS,ES)
    Sntlinit:
        EMIT          'S'
35        PUSH        SI
        PUSH        DI
        PUSH        CS
        POP          DS
        ASSUME DS:SNTL_INIT_SEG
40        XOR          BX, BX          ; copy original MBR over the
    subloader
        MOV          ES, BX          ; at location 0000:7C00h
        MOV          DI, 7C00h
45        MOV          AX, OFFSET part_sector
        MOV          SI, AX          ; SI = sector to copy.
        MOV          CX, 100h        ; 256 words to copy.
        CLD
        REP MOVSW
50        EMIT          'M'
    ; Check if sntlinit is already in memory.
        XOR          BX, BX
        MOV          ES, BX
        MOV          BX, [ES:00BCh]
55        MOV          ES, [ES:00BEh]
        CMP          [WORD ES:BX+3], 'PR'
        JNZ          RPL_check_fail
        CMP          [BYTE ES:BX+5], 'L'
        JNZ          RPL_check_fail
60 RPL_exist:
    ; Check if the sentinel acknowledges.
        EMIT          'R'
        MOV          AX, SNTL_SIG1
        MOV          DX, SNTL_SIG2
65        INT          2Fh

```

SUBSTITUTE SHEET

```

    CMP     DX, SNTL_SIG2      ; Is the sentinel installed?
    JNE     exit              ; Yes, exit now.
    EMIT    '!'

5   RPL_check_fail:

        XOR     AX, AX
        MOV     ES, AX        ; ES = IVT segment.
    ; Calculate and assign SNTL_SEG to DS.
10        MOV     AX, CS
        MOV     BX, OFFSET SNTL_SEGMENT
        SHR     BX, 4
        ADD     AX, BX
        MOV     DS, AX
15        ASSUME DS:SNTL_SEG

    ; Hook the interrupt handlers into the system:
        CLI                      ; DISABLE INTERRUPTS
    ; Hook 2Fh.
20        MOV     [WORD ES:00BCh], OFFSET Int2FVect
        MOV     [WORD ES:00BEh], AX
    ; Hook 13h.
        MOV     AX, [WORD ES:004Ch] ; first hook of INT
25    13h to control disk access
        MOV     [WORD sngprvds1], AX
        MOV     AX, [WORD ES:004Eh]
        MOV     [WORD sngprvds1+2], AX
        MOV     [WORD ES:004Ch], OFFSET Int13ISR
30        MOV     [WORD ES:004Eh], DS

;*****
;      MOV     AX, [WORD ES:0064h]      ; hook INT 19h to track
reboot
35    ;      MOV     [WORD sngprvint19], AX
;      MOV     AX, [WORD ES:0066h]
;      MOV     [WORD sngprvint19+2], AX
;      MOV     [WORD ES:0064h], OFFSET snfint19
;      MOV     [WORD ES:0066h], DS
;
40    ;
;      MOV     [BYTE ES:03C4h], 'N'      ; QEMM requirement
DOSDATA look like ; to work with QEMM
;      MOV     [BYTE ES:03C5h], 'e'      ; a Novell NetWare RPL by
45    ;      MOV     [BYTE ES:03C6h], 't'      ; this string at INT F1h
and our code      ;
;      MOV     [BYTE ES:03C7h], 'w'      ; segment (less DOS
wrapper) at
50    ;
F3h                ; segment portion of INT
;
;      MOV     AX, DS                    ; Novell puts its INT 13h
address at
;      SUB     AX, 0001h                 ; INT F3h, so try that for
55    our hook
;      MOV     [WORD ES:03CCh + 2], AX
;      MOV     [WORD ES:03CCh], OFFSET Int13ISR
;*****

60    ; Initialize runtime variables (if any).
        MOV     AX, [modem_default_port] ; set first port to attempt
dial out
        MOV     [sngmdmprt], AX
; Set the load time variable for the preint12_handler.
65    ;      MOV     AX, 0040h

```

SUBSTITUTE SHEET

```

;          MOV     ES, AX
; time.      MOV     AX, [ES:006Ch]          ; ES = bios segment.
;          ; Load current bios
5      MOV     [load_time], AX

      STI
;          ; ENABLE INTERRUPTS.

      EMIT     'H'

10     exit:
; Jump to io.sys
      EMIT     'X'
      POP     DI
      POP     SI
15     ASSUME ES:NOTHING
      POP     ES
      ASSUME DS:NOTHING
      POP     DS
      POP     DX
20     POP     CX
      POP     BX
      POP     AX
; Jump back to sector.
25     JmpOpcode DB 0EAh
      EntryPnt  DW 7C00h
      SectSeg   DW 0000h

; IF EMIT_ON ;Only needed for EMIT macro -----
;
30     ; Puts the character in AL to the console.
;
PROC PutChar NEAR
      PUSH     AX
      PUSH     BX
35     MOV     AH, 0Eh          ;Output a character
      MOV     BH, 0
      push     bp              ;TCN - For old BIOS
      INT     10h
      pop      bp              ;TCN - For old BIOS
40     POP     BX
      POP     AX
      RETN
ENDP PutChar

45     ENDIF ;EMIT_ON-----

; Padding to maintain segment offset that matches the current CTM.EXE
      Padding DB 20h DUP( 90h )

50     ;Following statments must be at the end of the SNTL_INIT_SEG.
      ALIGN 16
SNTL_SEGMENT:
of SNTL_SEG.          ; Used to calculate the location
ENDS

55     END

```

SUBSTITUTE SHEET

```
*****
;*****
; Copyright (c) Absolute Software 1994, 1995
;
5 ; SNTLJTBL.ASM
;
; Contains the main jumtable code used by TimerISR.
;
; HISTORY:
10 ; 1995.09.05 - CCOTI
; Created.
;
;*****
*****

15 IDEAL

;NOLIST
include "SENTINEL.INC"
20 include "SNTLJTBL.INC"
include "SNTLDATA.INC"
include "SNTLTIMR.INC"
;LIST

25 SEGMENT SNTL_SEG BYTE PUBLIC 'CODE'
ASSUME CS:SNTL_SEG, DS:SNTL_SEG, ES:NOTHING
;
;
; Enter: AL = table index, BX = table offset.
30 PROC JumpTable NEAR
XOR AH,AH ; zero AH
SHL AX,1 ; multiply AX by 2 to get
offset
ADD BX,AX ; add offset to the table base
35 JMP [WORD BX] ; jump the indexed address
ENDP JumpTable

cleanup:
MOV [sngstftn],OFFSET snfenrst
40 ; MOV [cleanup_routine],OFFSET ActiveRoutine
MOV [Sentinel_state],SNSTACTIVE
RETN

find_timeout:
45 IF 1
MOV [sngstftn],OFFSET snfenrst
ELSE
MOV [sngstftn],OFFSET snfenrst
50 MOV [cleanup_routine],OFFSET CheckNextPort
ENDIF
RETN

find_ok:
; Modem successfully initialized.
55 MOV [BYTE init_str_num],INIT_STR_TABLE_SIZE ;reset modem
init string table index

init_error:
60 MOV [sngstftn],OFFSET ModemInitInit
RETN

init_ok:
IF 0
;MOD DBOYD 55:95.04.12
```

SUBSTITUTE SHEET


```

; reset the dial string number when the Sentinel goes active, doing
; this will
; allow the system to search for another port and continue on from
; the last
5  ; pre-dial string used
    MOV      [BYTE dial_str_num],DIAL_STR_TABLE_SIZE
ENDIF
    MOV      [sngstftn],OFFSET ModemCallInit
10    RETN

IF 0
;MOD DBOYD 50:95.02.22:
; to allow direct dial and PBX dial to work on opposite system,
; treat <BUSY>
15 ; the same as <NO_DIAL_TONE> so that the next pre-dial string will
    be used
    dial_busy:
        DEC      [dial_str_num]          ; reuse the last
    dial_string
20    MOV      [sngstftn],OFFSET ModemCallInit
    RETN
ENDIF

IF 1
25 dial_busy:          ;MOD DBOYD 50:95.02.22
    dial_error:      ;MOD DBOYD 55:95.04.12
    dial_no_carr:    ;MOD DBOYD 55:95.04.12
    cnct_error:      ;MOD DBOYD 55:95.04.13
ENDIF
30 dial_no_tone:
    MOV      [sngdlytmr], TM1SEC          ; delay 1 second before
    redialing
    MOV      [sngstftn], OFFSET ModemFindInit ; search for modem
    before redialing
35    RETN

IF 0
;MOD DBOYD 55:95.04.12:
40 ; to allow 8 prefix to work on 9 prefix PBX's and direct dial, treat
; the responses below the same as no dial tone so that the next pre-
; dial
; string will be used
    dial_error:
45    dial_no_carr:
ENDIF
    cmrxpktto:          ;ADD CCOTI 48:95.02.07
    MOV      [sngstftn],OFFSET snfsnrst
    ;
50    MOV      [cleanup_routine],OFFSET ActiveRoutine
    MOV      [sngclst], SNCALLFAIL
    RETN

IF 0
;MOD DBOYD 55:95.04.13
55 cnct_error:
ENDIF
    cnct_eot:
    MOV      [sngstftn],OFFSET snfsnrst
    ;
60    MOV      [cleanup_routine],OFFSET ActiveRoutine
    RETN

dial_server:
cnct_eng:
cnct_nak:
65 cnct_resend:
    MOV      [sngstftn],OFFSET snftxchkin
```

SUBSTITUTE SHEET

```

                                RETN
cnct_ack:
5      MOV      (angstftn),OFFSET anfgetpkt
      RETN

      IF 0
cnct_hold:
10     MOV      [receive_tick_count],0
      timeout.
      RETN
      ; Reset
ENDIF
15     ENDS
      END
```

SUBSTITUTE SHEET

```

;*****
;*****
; * Copyright (c) Absolute Software 1994, 1995
; *
5  ; * SNTLSTRT.ASM
; *
; * Contians routines for using the sentinel's string tables.
; *
; * HISTORY:
10 ; *   1995.09.05 - CCOTI
; *       Created.
; *
;*****
;*****
15
IDEAL

%NOLIST
include "SENTINEL.INC"
20 include "SNTLSTRT.INC"
include "SNTLDATA.INC"
include "SNTLBUFF.INC"
%LIST

25 SEGMENT SNTL_SEG BYTE PUBLIC 'CODE'

;*****
;*****
30 ;
; COMTRANSCHECK - check result of transmission
;
; PURPOSE:
;   This function checks the result of a transmission between the
Sentinel
35 ;   and the modem. It test modem responses against a table of
strings. More
;
; PARAMETERS:
;   BX = the beginning of the string table (more than one table is
40 supported)
;
; RETURNS:
;   CF = 1 if response is not completely received
;   CF = 0 and AL = 0 if time-out without a match
45 ;   CF = 0, AL = index of match in response table (1 = last string
in table)
;
; GLOBALS REFERENCES:
;   receive_tick_count
50 ;   sngrxbuftl
;   sngrxbufhd
;
; GLOBALS MODIFIED:
;   sngrxbuftl - if a string is found, set past the found string.
55 ;
; BIOS CALLS:
;   None
;
; DOS CALLS:
60 ;   None
;
; PROCEDURE CALLS:
;   buf_inc_ptr, buf_getchar
;
65 ; HARDWARE ACCESS:

```

SUBSTITUTE SHEET

```

;      None
;
;      NOTES:
5      ;
;.....
;.....

      ASSUME CS:SNTL_SEG, DS:SNTL_SEG, ES:NOTHING

10     PROC ComTransCheck

            CLD
increment    ; set CMPSB pointers to
            PUSH    DS
            POP     ES
            MOV     AL, [BX-1]
            XOR     CH, CH
            XOR     AH, AH
            ; get ES = DS
            ; AL = the number of strings
            ; zero CH (CL is defined
            ; make AH zero

@@str_loop_start:
; Initialize the inner loop.
            MOV     SI, [engrxbuf1]
            MOV     DI, [BX]
            ADD     BX, 2
            MOV     CL, [DI-1]
            ; DI = the string to check
            ; CX = the string len

30     @@char_loop_start:
            CMP     SI, [engrxbufhd]
            JE      @@buffer_ overrun
            CMPSB
            JNE     @@unmatched_byte
            LOOP     @@char_loop_start
            ; at the end of the buffer
            ; this increments SI and DI
            ; this string doesn't match

35     @@found match:
            MOV     [engrxbuf1], SI

40     @@clear_carry:
            CLC
            RET
            ; set return status
            ; exit

@@buffer_ overrun:
            INC     AH
45     found      ; AH != 0 if no match has been

@@unmatched_byte:
; Have we checked all of the strings?
            DEC     AL
            JNZ     @@str_loop_start
            ; decrement string counter

50     @@no_match:
            ; Check if we have timed out.
            CMP     [rx.rxtmr], 0
            JE      @@clear_carry
            ; AL = 0
            ; timed out, exit

; Check if we need to consume a character.
            OR      AH, AH
            ; AH != 0 if no match was

60     found      JNZ     @@exit
            CALL    buf_getchar

@@exit:
65     STC
            RET
            ; set return status
            ; exit

```

SUBSTITUTE SHEET

ENDP ComTransCheck
ASSUME NOTHING

5

ENDS

END

SUBSTITUTE SHEET

```
;*****
*****
;* Copyright (c) Absolute Software 1994, 1995
;*
5  ;* SNTLTIMR.ASM
;*
;* Contains Timer ISR and related subroutines.
;*
;* HISTORY:
10  ;*   1995.09.05 - CCOTI
;*       Created.
;*
;*****
*****
15  IDEAL

    \NOLIST
    include "SENTINEL.INC"
20    include "SNTLTIMR.INC"
    include "SNTLDATA.INC"
    include "SNTLJTBL.INC"
    include "SNTLAPI.INC"
    include "SNTLCOMM.INC"
25    include "SNTLCOMV.INC"
    include "SNTLBUFF.INC"
    include "SNTLSTRT.INC"
    \LIST

30    ;TCN Nov 1/95
    MACRO TCN_EMIT ch
        PUSH    AX
        MOV     AL,ch
35        CALL   TCN_PutChar
        POP     AX
    ENDM
    ;TCN Nov 1/95

40    SEGMENT SNTL_SEG BYTE PUBLIC 'CODE'
        ASSUME_CS:SNTL_SEG, DS:NOTHING, ES:NOTHING

    ;TCN Nov 1/95
    PROC TCN_PutChar NEAR
45        PUSH    AX
        PUSH    BX
        MOV     AH,0Eh                ;Output a character
        MOV     BH,0
        push    bp                    ;TCN - For old BIOS
50        INT     10h
        pop     bp                    ;TCN - For old BIOS
        POP     BX
        POP     AX
        RETN
55    ENDP TCN_PutChar
    ;TCN Nov 1/95

;*****
*****
60    ;*
;* TMFISR - timer interrupt service routine
;*
;* PURPOSE:
65    ;*   This function implements the timer ISR that is hooked to the
    system
```

SUBSTITUTE SHEET

```

/* timer. This function and performs the following:
/*
/* Checks the Sentinel's state <Sentinel state>
5 /* Executes one of the following subroutines based on the state:
/*   SNSTACTIVE
/*     ActiveRoutine:
/*
/*   SNSTALERT
10 /*     PortFindInit:
/*       PortFind
/*       CheckNextPort:
/*
/*   SNSTCALL
15 /*     ModemFindInit:
/*       ModemFind:
/*       ModemFindError:
/*
/*     ModemInitInit:
/*     ModemInit:
20 /*     ModemInitError:
/*     ModemCallInit:
/*     ModemCall:
/*     ModemDialError:
/*
25 /*   SNSTCONNECT
/*     snftxchkin:
/*     ModemConnect:
/*     ModemConnectError:
/*
30 /*   SNSTERROR
/*     ErrorRoutine:
/*
/* PARAMETERS:
/*   None
35 /* RETURNS:
/*   Nothing
/*
/* REGISTERS DESTROYED:
40 /*   None
/*
/* GLOBALS REFERENCED:
/*   engincmiser
45 /*   Sentinel_state
/*   engstftn
/*   time_count
/*   activation_period
/*   modem_default_port
/*   port_table
50 /*   engmdmprtadd
/*   engmdmprtint
/*   modem_init_str
/*   init_result_table
/*
55 /* GLOBALS MODIFIED:
/*   engmdmprt - set to the port currently being used.
/*   Sentinel_state - set to the now state of the Sentinel
/*   engstftn - set to the routine that will perform the next
60 action.
/*   engmdmprtadd - set to the address used by the current port
/*   (engmdmprt)
/*   engmdmprtint - set to the interrupt used by the current port
/*   (engmdmprt)
65 /*   engincmiser - reset to 0 before cmfiser is hooked in.
/*   send_buf_len - reset before cmfiser is hooked in.
```

SUBSTITUTE SHEET

```

;*      sngprvcom - stores the old com ISR before hooking in cmfiser.
;*
;*
5  ;* BIOS CALLS:
;*      None
;*
;* DOS CALLS:
;*      None
;*
10 ;* PROCEDURE CALLS:
;*      buf_flush
;*      SwapInt
;*      ComTransInit
;*      ComTransCheck
15 ;*
;* HARDWARE ACCESS:
;*      UART (I/O MCR, OUT IER), I/O PIC
;*
20 ;*****
;*****

PROC tmfiser FAR                                ; Entry point for TimerISR.

25 IF Debug
    INC [CS:TimerISR_count]                    ; increment debug timer
ENDIF

    CMP [CS:sngsuspend], 0                      ; is the Sentinel
30     JNE TimerAbort                          ; yes, exit
    CMP [CS:sngincmiser], 0                    ; is the Sentinel in the
    JNE TimerAbort                          ; yes, exit
35     PUSH DS
    PUSH ES
    PUSHA
    ; save registers

40     PUSH CS
    SNTL_COM_SEG
    POP DS
    ASSUME DS:SNTL_SEG
    ; set DS = CS =

45     CLI
    CMP [sngcomhk], 0                          ; halt interrupts
    JNE @tmcont                                ; have we hooked the comm.
    ; no, continue

50     ; yes, determine if we
    still have the hook
    MOV BX, [sngmdmptint]                      ; the IVT entry to test
    SHL BX, 2                                  ; BX = the IVT offset to
55     get ISR vector
    XOR AX, AX
    MOV ES, AX
    ; clear ES

    MOV AX, [ES:BX]                            ; get offset of installed
60     vector
    CMP AX, OFFSET cmfiser                     ; is it our routine?
    JNE @tmrst                                ; no, Reset sentinel
    MOV AX, DS
    CMP AX, [ES:BX+2]                          ; get our segment
    ; compare to installed
    vector segment

```

SUBSTITUTE SHEET


```

JNE      @@tmrst      ; if not equal, reset
Sentinel

5      JMP      @@tmcont      ; we still have the
interrupt, continue

@@tmrst:      ; reset the Sentinel and
continue
10      MOV      [sngstftn], OFFSET ActiveRoutine
MOV      [Sentinel_state], SNSTACTIVE
MOV      [sngcomhk], 0      ; clear the the comm.
hooked flag

@@tmcont:
15      STI      ; restore interrupts
CMP      [win_flag], 0      ; are we in Windows?
JE      @@chktmrs      ; no, go check running
timers
20      MOV      AX, 1683h      ; yes, determine virtual
machine
INT      2Fh
CMP      BL, [win_vm]      ; should be a word!!
JNE      TimerExit      ; not in virtual machine 1,
exit
25      @@chktmrs:
CMP      [tx.txxtmr], 0      ; is the transmit timer
running?
30      JE      @@nxtmr0      ; no, continue
DEC      [tx.txxtmr]      ; yes, decrement the
transmit timer

@@nxtmr0:
35      CMP      [rx.rxxtmr], 0      ; is the receive timer
running?
JE      @@nxtmr1      ; no, continue
DEC      [rx.rxxtmr]      ; yes, decrement the port
delay timer
40      @@nxtmr1:
CMP      [sngprtdlytmr], 0      ; is the port delay timer
running?
45      JE      @@nxtmr2      ; no, continue
DEC      [sngprtdlytmr]      ; yes, decrement the port
delay timer

@@nxtmr2:
50      CMP      [sngdlytmr], 0      ; is the Sentinel delay
timer running?
JE      @@gostate      ; no, execute state
function
DEC      [sngdlytmr]      ; yes, decrement timer
JMP      TimerExit      ; no, call previous timer
handler
55      @@gostate:
CALL      [sngstftn]      ; execute the state
function

60      TimerExit:
POPA      ; recover registers
ASSUME DS:NOTHING, ES:NOTHING
POP      ES
65      POP      DS
```

SUBSTITUTE SHEET

```
TimerAbort:
    JMP     [DWORD CS:engprvtmr]

5  ENDP tmfier
    ASSUME NOTHING

;*****
;*****
10 ;*
;* SNFWTFORXMS - wait for XMS
;*
;* PURPOSE:
15 ;* This function waits for the extended memory manager (XMS) to be
loaded
;* and then hooks interrupt 2Fh. This hook allows the Sentinel to
track the
;* PC moving in and out of WINDOWS, and allows ASC utilities to
communicate
20 ;* with the utility.
;*
;* PARAMETERS:
;* None
25 ;*
;* RETURNS:
;* Nothing
;*
;* GLOBALS REFERENCED:
;*
30 ;* GLOBALS MODIFIED:
;*
;* BIOS CALLS:
;* None
35 ;*
;* DOS CALLS:
;* None
;*
;* PROCEDURE CALLS:
;* None
40 ;*
;* HARDWARE ACCESS:
;* Nothing
;*
45 ;*****
;*****

    ASSUME CS:SNTL_SEG, DS:SNTL_SEG, ES:NOTHING

50 ;*** STOLEN BY CCOTI ***

    ASSUME NOTHING

;*****
;*****
55 ;*
;* SNFWAIT - wait for timer to expire
;*
;* PURPOSE:
60 ;* This function waits for the delay timer, sngdlytmr, to expire
before
;* allowing the Sentinel to proceed. This function is used to
delay the
;* Sentinel from activating on power-up and when entering and
exiting
```

SUBSTITUTE SHEET

```

5  ;*   WINDOWS.  Since the delay may be started at any time for a
    number of
    ;*   reasons, when the delay expires the Sentinel goes to snfnrst()
    ;*   before going back to ActiveRoutine().
    ;*
    ;*   PARAMETERS:
    ;*       None
    ;*
10  ;*   RETURNS:
    ;*       Nothing
    ;*
    ;*   GLOBALS REFERENCED:
    ;*
15  ;*   GLOBALS MODIFIED:
    ;*
    ;*   BIOS CALLS:
    ;*       None
    ;*
20  ;*   DOS CALLS:
    ;*       NONE
    ;*
    ;*   PROCEDURE CALLS:
    ;*       None
    ;*
25  ;*   HARDWARE ACCESS:
    ;*       None
    ;*
    ;*****
    ;*****
30
    ASSUME CS:SNTL_SEG, DS:SNTL_SEG, ES:NOTHING

    IF 0
    PROC snfwait NEAR
35
        CMP        [sngdlytmr], 0                ; wait for delay timer to
    expire
        JNE        @@exit                        ; not yet expired, exit
40
    @@reset:
        MOV        [sngstftn],OFFSET snfnrst      ; reset the Sentinel
    ;        MOV        [cleanup_routine],OFFSET ActiveRoutine
45
    @@exit:
        RETN                                     ; exit
    ENDP snfwait

    ASSUME NOTHING
50  ENDIF

    ASSUME CS:SNTL_SEG, DS:SNTL_SEG, ES:NOTHING
55  PROC ActiveRoutine NEAR

    ; Check if the activation period has been exceeded.
    ;       ROR        [cycle_var],1
    ;       JNC        @@end
60  ; Get current date and time.
        MOV        AH,4
        INT        1Ah                          ; Get RTC date.
        JC         @@end
        MOV        [now_year],CL                ; Store year.
65  XCHG        DL,DH

```

SUBSTITUTE SHEET

```

MOV      [WORD now_month],DX      ; Store month and day.
MOV      AH,2
INT      1Ah
JC       @@end
5      XCHG      CL,CH
MOV      [WORD now_hour],CX      ; Store hour and minute.

; Check if next call date has been passed.
MOV      SI, OFFSET next_call_date
10     MOV      DI, OFFSET now_date
CALL     CmpDates
JNC      @@end

@@alert:
15     MOV      [Sentinel_state],SNSTALERT      ; Date passed, set
to alert.

@@end:
; Check if we've been activated.
20     CMP      [Sentinel_state],SNSTALERT      ; Check state.
JNE      @@exit

@@activated:
25     ; Set the Sentinel to the ALERT state.
IF 0
MOV      AX,[modem_default_port]
MOV      [sngmdmprt],AX      ; set first
30     port
ENDIF
MOV      [BYTE dial_str_num],DIAL_STR_TABLE_SIZE      ; set first
pre-dial string
MOV      [sngstftn],OFFSET PortFindInit      ; set next state
35     function

@@exit:
RETN
ENDP ActiveRoutine

40     ;
;
;
PROC CheckNextPort NEAR
45     MOV      AX,[sngmdmprt]
INC      AX
CMP      AL,PORT_TABLE_SIZE
JB       @@assign_port
XOR      AX,AX      ; start back at first port
50     @@assign_port:
MOV      [sngmdmprt],AX      ; set the modem port to
check
; go look for modem on the
port
55     MOV      [sngstftn],OFFSET PortFindInit
RETN      ; exit

ENDP CheckNextPort

60     ;
;
;
PROC PortFindInit NEAR
65     ; initialize PortFind variables (based on sngmdmprt which was set
previously).

```

SUBSTITUTE SHEET

```

MOV      [sngclst], SNPRTSRCH      ; set call status

MOV      BX, [sngmdmprt]
SHL      BX, 2
ADD      BX, OFFSET port_table
MOV      AX, [BX]
OR       AX, AX                    ; check if port is valid

JZ       CheckNextPort
10      MOV      [sngmdmprtadd], AX      ; store current port
address
MOV      AX, [BX+2]
MOV      [sngmdmprtint], AX          ; store current port
interrupt
15      MOV      [sngstftn], OFFSET PortFind ; set next state
MOV      [sngprtdlytmr], TM5SEC      ; set port delay timer to 5
seconds
RETN
ENDP PortFindInit

20
;
;
;
25      PROC PortFind NEAR
; Check if the port exists.
; ...
; NOT IMPLEMENTED - NEEDED FOR PCMCIA
; ...
30

; TCN_EMIT      '1'                      ;TCN Nov 1/95
35      MOV      DX, [sngmdmprtadd]      ; DX = current port's
address
INC      DX                          ; DX = current port's IER
IN       AL, DX                      ; AL = IER port status
IODELAY
40      AND      AL, 0FFh              ; if IER = 0xff, UART does
not exist
CMP      AL, 0FFh
JNE      @@chkprtavl                  ; port exists, go check
availability
JMP      CheckNextPort                ; port does not exist, go
45      check next port

@@chkprtavl:                          ; check if the port is in
use

50      ; TCN_EMIT      '2'                      ;TCN Nov 1/95

MOV      CX, [sngmdmprtint]          ; test PIC IMR first
SUB      CL, 08h                     ; get bit of interest
55      MOV      BL, 01h
SHL      BL, CL                      ; bit mask ready

IN       AL, 21h                     ; get primary PIC IMR
IODELAY
60      AND      AL, BL                ; bit set => interrupt
disabled
JNZ      @@port_idle

65      ; TCN_EMIT      '3'                      ;TCN Nov 1/95

```

SUBSTITUTE SHEET

```

next                                ; PIC IMR bit set, test IER
MOV DX,[sngmdmpprtadd]              ; DX = current port's
address                             ; DX = current port's IER
5   INC DX                          ; AL = IER port status
    IN AL,DX
    IODELAY
    OR AL,AL
10  JZ @@port_idle                  ; Are any IER bits set?
                                        ; if no, port is idle
    ; TCN_EMIT '4'
                                        ;TCN Nov 1/95

15  bits                           ; PIC IMR bit set, and IER
MOV DX,[sngmdmpprtadd]              ; set, test OUT2 next
address                             ; DX = current port's
20  ADD DX,MCR                      ; DX = current port's MCR
    IN AL,DX                        ; AL = MCR port status
    IODELAY
    TEST AL,08h                     ; is MCR OUT2 bit set?
    JZ @@port_idle                  ; if no, port is idle
25  ; TCN_EMIT '5'
                                        ;TCN Nov 1/95
    JMP CheckNextPort               ; all checks failed

30  @@port_idle:
    ; TCN_EMIT '6'
                                        ;TCN Nov 1/95
    CMP [sngprtdlytmr], 0           ; port must be available
35  for a set period
    JNE @@exit                       ; before a call is
    attempted

40  eight                           ; set port for no parity,
MOV DX,[sngmdmpprtadd]              ; data bits, and 1 stop bit
ADD DX,LCR                          ; get address of LCR
MOV AL,00000011b                    ; set LCR for N81
45  OR AL,80h                       ; set DLAB
    OUT DX,AL                       ; set value in LCR
    IODELAY

50  ; force 9600 bps
    ; DX = f / ( 16 * bps )
    ; = 1.8432 MHZ ( 16 *
    ; = 0x000C
55  MOV DX,[sngmdmpprtadd]          ; get address of DL LSB
    ADD DX,BRDl                     ; set new divisor
    MOV AX,000Ch
    OUT DX,AX
    IODELAY
60  MOV DX,[sngmdmpprtadd]          ; get address of DL LSB
    ADD DX,LCR                      ; get value in LCR
    IN AL,DX
    IODELAY
65  AND AL,7Fh                      ; clear DLAB

```

SUBSTITUTE SHEET

```

        OUT      DX, AL                      ; set value in LCR
        IODELAY

5      @@init_ok:
        ; Clear any pending errors in the UART.
        MOV      DX, [sngmdmprtadd]          ; get address of LSR
        ADD      DX, LSR
        IN       AL, DX
        IODELAY

10     ; Hook into the port, first init and install the interrupt vector.
        CALL     buf_flush                    ; flush the receive buffer
        MOV      [sngincmiser], 0
        MOV      [send_buf_len], 0
15     MOV      BX, [sngmdmprtint]            ; The int to install.
        MOV      DI, OFFSET sngprvcom         ; DS:DI = the address to
store the old vect.
        MOV      CX, OFFSET cmfiser          ; DS:CX = the new com
20     vector.
        CALL     SwapInt
        MOV      [sngcomhk], 1                ; set the comm. hooked flag

25     CLI
        MOV      DX, [sngmdmprtadd]          ; disable interrupts
        ADD      DX, MCR                      ; get address of MCR
        IN       AL, DX
        IODELAY
30     OR        AL, 00001011b
        OUT      DX, AL                      ; interrupts enabled in the
UART
        IODELAY

35     MOV      CX, [sngmdmprtint]            ; clear (enable) IRQ bit
mask in PIC
        SUB      CL, 08h
        MOV      BL, 01h
        SHL      BL, CL
40     NOT      BL
        IN       AL, 21h
        IODELAY
        AND      AL, BL
        OUT      21h, AL                    ; interrupts enabled in the
45     PIC
        IODELAY

        MOV      DX, [sngmdmprtadd]          ; get address of IER
        INC      DX
50     MOV      AL, 00000001b                ; interrupt when data
received
        OUT      DX, AL
        IODELAY

55     STI                                  ; enable interrupts

        MOV      [Sentinel_state], SNSTCALLING
        MOV      [sngdlytmr], TM1SEC         ; delay 1 second before
attempting to
        MOV      [sngstftn], OFFSET ModemFindInit ; find modem

60     @@exit:
        RETN
ENDP PortFind

65

```

SUBSTITUTE SHEET

```

;
;
;
5  PROC ModemFindInit NEAR
    MOV     [sngclst], SNMDMSRCH           ; set call status
    MOV     BX, OFFSET modem_find_str      ; get a pointer to modem
10  string  CALL cmfprpmdm                  ; prepare transmit
    structure MOV [tx.txxxxtst], OFFSET ModemFind ; set next state after
15  transmission
    RETN
    ENDP ModemFindInit
20
;
;
;
25  PROC ModemFind NEAR
    MOV     BX, OFFSET command_result_table ; check for received data
    CALL    ComTransCheck
30  JC      @@end                          ; data not received yet
    MOV     BX, OFFSET find_jump_table     ; check for acceptable
    response
    mov     [sngdlytmr], 9                 ;TCN Nov 1/95
35  spec.                                ;According to Hayes Modem
    0.5 secs                             ;we should wait at least
    command                               ;after sending the "ATZ"
40  JMP     JumpTable
    @@end:
    RETN
45  ENDP ModemFind
;
;
;
50  PROC ModemInitInit NEAR
    ; Attempt to initialize the modem (send modem_init_str).
55  MOV     [sngclst], SNMDMINIT           ; set call status
    MOV     BX, OFFSET init_str_num        ; get the index of the next
60  string  DEC [BYTE BX]
    JZ      @@reset                        ; wrap-around and start
    over
65  structure                             ; prepare transmit
```

SUBSTITUTE SHEET


```

    MOV     AX, [BX]                                ; get a pointer to the next
string
    SHL     AX, 1
    ADD     BX, AX
5      MOV     BX, [BX]
    CALL    cmfprpmdm                                ; prepare transmit
structure
    MOV     [tx.txxxxtst], OFFSET ModemInit          ; set state following
10      transmission
    RETN

@@reset:
15      MOV     [BYTE BX], INIT_STR_TABLE_SIZE      ; retry initialization
string
    MOV     [sngstftn], OFFSET ModemCallInit
    RETN

20      ENDP ModemInitInit

;
;
;
25      PROC ModemInit NEAR
; Check for reply.

30      MOV     BX, OFFSET mdm_init_result_table
    CALL    ComTransCheck
    JC      @@end
received yet                                ; data not

35      MOV     BX, OFFSET init_jump_table
    JMP     JumpTable

@@end:
    RETN
40      ENDP ModemInit

;
;
;
45      PROC ModemCallInit NEAR
; Attempt to dial (send modem pre-dial string).

50      MOV     [sngclst], SNMDMPD                    ; set call status

@@getstr:
    MOV     BX, OFFSET dial_str_num                  ; get the index of the next
string
    DEC     [BYTE BX]
55      JZ      @@reset
over                                         ; wrap-around and start

    MOV     AX, [BX]
    SHL     AX, 1
    ADD     BX, AX
60      MOV     BX, [BX]
    CALL    cmfprpmdm                                ; prepare transmit
structure
    MOV     [tx.txxxxtst], OFFSET ModemCallInit2    ; set state following
65      transmission
```

SUBSTITUTE SHEET

```

    RETN

@@reset:
    MOV     [BYTE dial_str_num], DIAL_STR_TABLE_SIZE
    JMP     @@getstr
    RETN

    ENDP ModemCallInit

;
;
;
    PROC ModemCallInit2 NEAR

        MOV     [sngclst], SNMDMDL                ; set call status
        MOV     BX, OFFSET dial_number            ; get the packet length
        CALL    cmfprpmdm                        ; prepare transmit
    structure
        MOV     [tx.txxnxtst], OFFSET ModemCall ; set state following
    transmission
                                                ; override default response
        time and
        MOV     [rx.rxtmr], TM40SEC                ; wait 40 seconds for
    response
    RETN

    ENDP ModemCallInit2

;
;
;
    PROC ModemCall NEAR

        MOV     [sngclst], SNWTCON                ; set call status
        MOV     BX, OFFSET dial_result_table
        CALL    ComTransCheck                    ; Check for reply.
        JC      @@end                             ; Data not received yet.

        MOV     BX, OFFSET dial_jump_table        ; attempt to parse data
        JMP     JumpTable

    @@end:
    RETN

    ENDP ModemCall

;
;
;
    PROC snftxchkin NEAR

; Query from server received by this point, send data packet

    structure
        MOV     AL, [sngdatalen]                ; get the data segment
    length
        ADD     AL, 2                            ; add 2 for type and
    subtype

```

SUBSTITUTE SHEET

```

5      MOV    [BYTE LOW tx.txxpktlen], AL
      MOV    [BYTE HIGH tx.txxpktlen], 0
      MOV    [tx.txxbufp], OFFSET sn_data_start
      MOV    [tx.txxnxtst], OFFSET snfgetpkt ; set state following
transmission
      MOV    [tx.txxpkttyp], CMTXDATPKT      ; transmitting data packet
      MOV    [tx.txxtmr], TM3SEC             ; set transmission timeout

10     MOV    [sngstftn], OFFSET cmftx       ; next state: transmit
      MOV    [rx.rxxtmr], TM10SEC           ; wait 10 seconds for
response
      MOV    [rx.rxxstate], OFFSET cmfpack   ; receiver state: process
expected ACK

15     RETN

      ENDP snftxchkin

20     ;*****
      ;*****
      ;
      ; SNFGETPKT - collect packet data
      ;
25     ; PURPOSE:
      ; This functions collects packet data and determines if a receive
timeout
      ; has occurred.
      ;
30     ; PARAMETERS:
      ; None
      ;
      ; RETURNS:
      ; Nothing
35     ;
      ; NOTE:
      ;
      ;*****
      ;*****

40     PROC snfgetpkt NEAR

      MOV    [sngclst], SNWTNCD              ; set call status
      CMP    [rx.rxxtmr], 0                  ; test for timeout
45     JE     @@timeout                       ; timed out

      CALL    buf_getchar                     ; retrieve a character
      JC      @@exit                          ; none available, exit

50     CALL    [rx.rxxstate]                  ; run the rx state function
      RETN

@@timeout:
      MOV    [sngstftn], OFFSET cmftx       ; set next Sentinel state
55     function
      MOV    [tx.txxpkttyp], CMTXDLENQ      ; set transmitter state:
send ENQ

60     @@exit:
      RETN

      ENDP snfgetpkt

65

```

SUBSTITUTE SHEET

```

;
;
;
5  PROC snfenrst NEAR
; Reset the Sentinel to a known state (ACTIVE), assume nothing.
      CALL    buf_flush
      CMP     [sngcomhk], 1                ; have we hooked the comm.
port   JNE     @@cont                      ; no, continue
10
                                     ; yes, unhook the com
      interrupt
      MOV     BX,[sngmdmprtint]            ; the interrupt to install
      XOR     DI,DI
15     PUSH    DS
      LDS     CX,[sngprvcom]              ; DS:CX = the com vector to
install.
      CALL    SwapInt
      POP     DS
20     MOV     [sngcomhk], 0                ; clear the comm. hooked
flag
@@cont:
      MOV     DX,[sngmdmprtadd]
25     INC     DX                          ; DX = IER
      XOR     AL,AL
      OUT     DX,AL                        ; Disable all interrupts.
      IODELAY
      ADD     DX,MCR-IER                   ; DX = MCR
30     OR      AL,03h                      ; leave RTS & DTR asserted
to get <NO CARRIER>
      OUT     DX,AL                        ; MCR OUT2 bit = 0
      IODELAY
35     MOV     [sngstftn], OFFSET ActiveRoutine
      RETN
ENDP snfenrst
40  ENDS
      END
```

SUBSTITUTE SHEET

Electronic Article Surveillance System
Source Code for Host-side
Visual C++ (Microsoft)

```
5  /*=====*\
   Description:
      Source code for CompuTrace Server and DBServer.
10  Copyright:
      Copyright 1993-1995 Absolute Software Inc. All
      Rights Reserved.
15  \*=====*/

#define INCL_NOPMAPI           // no PM in this program.
#define INCL_DOS
20  #define INCL_BSE
    #include <os2.h>
    #include <fstream.h>
    #include <time.h>

25  #include <server.h>
    #include <DB_Objects.HPP>
    #include <CTMessage.HPP>
    // #include <packet.h>
    #include "CT_Trans.H"
30

    FLAG fQueryCTIDStatus( MessagePipe &Pipe, const
    QueryCTIDStatusMsg &Status, CTIDStatusResultMsg &Result
    );
    FLAG fStoreMonitorEvent( MessagePipe &Pipe, const
35  StoreMonitorEventMsg &Store, StoreResultMsg &Result );
    FLAG fSignalQuit( MessagePipe &Pipe );

    void AssignTS( TTimestamp &ts, const SNTL_DATE &Date );
    void AssignSNTL_DATE( SNTL_DATE &Date, const TTimestamp
40  &ts );

    // Temp function.
    void ProcessClient( TPort &Port, TConnectInfo
    &ConnectInfo, CTID_TEXT *text );
45

    extern MessagePipe *pipe;

    //
    // SntlConnect: called when a CONNECT comand has been
50  received, this function processes
```

SUBSTITUTE SHEET

```
//          a transaction between the server and a
// Sentinel client.
//
5 void SntlConnect( TPort &Port, MessagePipe &Pipe,
TConnectInfo *cnct_info )
{
    WORD msg_type;

10    DosGetDateTime( &cnct_info->start_time );           //
    Fill start time.

    TPacket packet( Port );

    while (TRUE) {
15        // Get a packet.
        if (packet.rGetPacket() != TPacket::TRANS_ACK) {
            cout << "Packet Error" << endl;
            return;
        }
20        // Determine packet type.
        packet.cbCopyText( &msg_type, sizeof( msg_type ) );
        switch( msg_type ) {
            case CTID_TEXT_TYPE:
                // Create a new client object.
25                // TClient Client( Port, Pipe, *cnct_info );
                // Get CTID Text and add to Client object.
                CTID_TEXT Text;
                packet.cbCopyText( &Text, sizeof( Text ) );
                // Client.SetCTID( Text );
30                // ProcessClient.
                // ProcessClient( Client );
                ProcessClient( Port, *cnct_info, &Text );
                return;
            default:
35                return;
        }
    }
}

40 void ProcessClient( TPort &Port, TConnectInfo
&ConnectInfo, CTID_TEXT *text )
{
    SNTL_DATE next_call;

45    // ENTER APPLICATION LAYER...

    // Query the Client state.
    QueryCTIDStatusMsg StatusMsg;
    StatusMsg.CTID = (ULONG)text->sn[0] + ((ULONG)text->
50    >sn[1] << 16);

    CTIDStatusResultMsg Result;

    55    cout << "QueryCTIDStatus for CTID " << StatusMsg.CTID
    << "... ";
```

SUBSTITUTE SHEET

```
if (!fQueryCTIDStatus( *pipe, StatusMsg, Result )) {
    cout << "Error in QueryCTIDStatus!" << endl;
}
else {
5     cout << "CTIDStatusResult Received..." << endl;
    cout << "    Status = " << (STRING)Result.Status <<
endl;
    cout << "    PeriodDays = " << Result.PeriodDays <<
10 endl;
    cout << "    PeriodMinutes = " <<
Result.PeriodMinutes << endl;
    cout << "    StolenFlag = " <<
(STRING)Result.StolenFlag << endl;
    cout << "    SpecialProcess = " <<
15 Result.SpecialProcess << endl;
    cout << "    Orgnum = " << Result.Orgnum_n << endl;
}

// Send NextCall Message back to the Client.
20 CTTimestamp next_ts;
AssignTS( next_ts, text->now_date );
if (next_ts.usYear() < 1900) { // If date is not
valid substitute the local date instead.
25     next_ts = ConnectInfo.start_time;
}
next_ts.AddToDate( 0, 0, Result.PeriodDays, 0,
Result.PeriodMinutes );
AssignSNTL_DATE( next_call, next_ts );

30 SendDatePacket( Port, next_call );
SntlDisconnect( Port, ConnectInfo );

// Store the Monitor Event.
35 StoreMonitorEventMsg Event;
Event.StoreAsStolen = Result.StolenFlag;
Event.StoreAsExpire = FALSE;

Event.LicenseStatus = Result.Status;
AssignTS( Event.ClientTS, text->now_date );
40 Event.ServerTS = ConnectInfo.start_time;
Event.NextCallTS_n = Event.ServerTS;
Event.NextCallTS_n.AddToDate( 0, 0, Result.PeriodDays,
0, Result.PeriodMinutes );
Event.NextCallClientTS_n = next_ts;
45 Event.CTID = StatusMsg.CTID;
Event.TelcoTS_n.Assign( Event.ServerTS.usYear(),
                        ConnectInfo.cnd.month,
                        ConnectInfo.cnd.day,
                        ConnectInfo.cnd.hour,
50                        ConnectInfo.cnd.minute );

Event.DurationSec_n = 0;
Event.CallerID_n = (const
char(*)[CALLERID_SIZE])ConnectInfo.cnd.number;
55 Event.LineNum = 1;
Event.LogFlag = FALSE;
```

SUBSTITUTE SHEET

```
Event.EnvironmentID = "DBS-9508";
Event.ErrorCnt = 0;

StoreResultMsg ResultMsg;

5   cout << endl << "Storing the MonitorEvent... ";

   if (!fStoreMonitorEvent( *pipe, Event, ResultMsg )) {
10      cout << "Error in StoreMonitorEvent!" << endl;
   }
   else {
      cout << "StoreResult = " << (ResultMsg.Result ?
15      "TRUE" : "FALSE") << endl;
   }
}

void SendDatePacket( TPort& Port, const SNTL_DATE& date )
{
20   NC_PACKET packet;

   packet.header.stx = STX;
   packet.header.lsb_length = sizeof( NC_TEXT );
   packet.header.msb_length = 0;
25   packet.text.type = NC_TEXT_TYPE;
   packet.text.next_call_date = date;

   packet.footer.etx = ETX;
30   packet.footer.lrc = 0;

   Port.fWritePort( (PVOID)&packet, sizeof( packet ) );
}

35   FLAG fQueryCTIDStatus( MessagePipe &Pipe, const
   QueryCTIDStatusMsg &Status, CTIDStatusResultMsg &Result )
   {
      TStream in_strm, out_strm;
40      out_strm << Status;
      if (!Pipe.fTransact( out_strm, in_strm )) return
FALSE;
      in_strm >> Result;
45      if (Result.eType() == CTID_STATUS_RESULT) return TRUE;
      else return FALSE;
   }

50   FLAG fStoreMonitorEvent( MessagePipe &Pipe, const
   StoreMonitorEventMsg &Store, StoreResultMsg &Result )
   {
      TStream in_strm, out_strm;
55      out_strm << Store;
```

SUBSTITUTE SHEET


```
    if (!Pipe.fTransact( out_strm, in_strm )) return
FALSE;
    in_strm >> Result;

5      if (Result.eType() == STORE_RESULT) return TRUE;
      else return FALSE;
    }

10     FLAG fSignalQuit( MessagePipe &Pipe )
    {
        TStream stream;
        CliQuitMsg QuitMsg;

15         stream << QuitMsg;
        return Pipe.fSendMessage( stream );
    }

20     void SntlDisconnect( TPort &Port, TConnectInfo
    &ConnectInfo )
    {
        // Drop DTR.
        DosSleep( 500 );    // Broc - 13 Feb 95
25         // Add delay to let modem clear xmt
        buffer
        Port.fDropDTR();    // to fix intermittent modem fault.

30         cout << "Disconnecting..." << flush;

        DosGetDateTime( &ConnectInfo.end_time );    //
        Fill end time.
        DosSleep( 200 );

35         // Raise DTR.
        Port.fRaiseDTR();
    }

40     // *** helper functions.
    UCHAR BCD2ToUChar( BYTE bcd )
    {
        // Convert a two digit bcd number to decimal.
45         return (bcd >> 4) * 10 + (bcd & 0x0F);
    }

    BYTE UCharToBCD2( UCHAR dec )
    {
50         // Convert a 8 bit decimal number to bcd.
        return (dec % 10) + (((dec / 10) % 10) << 4);
    }

    USHORT BCD4ToUShort( WORD bcd )
55     {
```

SUBSTITUTE SHEET

```
// Convert a four digit bcd number to decimal.
return (bcd >> 12) * 1000 + ((bcd & 0x0F00) >> 8) *
100 + ((bcd & 0x00F0) >> 4) * 10 + (bcd & 0x000F);
}

5 WORD UShortToBCD4( USHORT dec )
{
// Convert a 16 bit decimal number to a 4 digit decimal.
return (dec % 10) + (((dec / 10) % 10) << 4) + (((dec
10 / 100) % 10) << 8) + (((dec / 1000) % 10) << 12);
}

void AssignTS( TTimestamp &ts, const SNTL_DATE &Date )
{
15     ts.Assign( BCD2ToUChar( Date.year ),
                BCD2ToUChar( Date.month ),
                BCD2ToUChar( Date.day ),
                BCD2ToUChar( Date.hour ),
20     BCD2ToUChar( Date.minute ) );
}

void AssignSNTL_DATE( SNTL_DATE &Date, const TTimestamp
&ts )
{
25     Date.year   = UCharToBCD2( ts.usYear() % 100 );
    Date.month   = UCharToBCD2( ts.usMonth() );
    Date.day     = UCharToBCD2( ts.usDay() );
    Date.hour    = UCharToBCD2( ts.usHour() );
30     Date.minute = UCharToBCD2( ts.usMinute() );
}

/*
inline BYTE HiNibble( BYTE b ) { return (BYTE)((b & 0xF0)
>> 4); }
35 inline BYTE LoNibble( BYTE b ) { return (BYTE)(b & 0x0F);
}

void AddDays( SNTL_DATE *next_call, int days )
{
40     static BYTE days_per_month[18] = {
        0x31,
        0x28,
        0x30,          // 0x03 - March
        0x31,
        0x30,
45     0x31,          // 0x06 - June
        0x30,
        0x31,
        0x30,          // 0x09 - Sept
        0x00,          // 0x0A
        0x00,          // 0x0B
        0x00,          // 0x0C
        0x00,          // 0x0D
50     0x00,          // 0x0E
        0x00,          // 0x0F
55
```

SUBSTITUTE SHEET

```

    0x31,      // 0x10 - Oct
    0x30,
    0x31      // 0x12 - Dec
};

5   BYTE old_day = next_call->day;
    // Save for BCD adjust.

10  // Add the days to the current date.
    next_call->day += days;
    // Check if we passed the end of the current month.
    if (next_call->day > days_per_month[next_call->month])
    {
15      // Add one to month.
        if (++next_call->month > 12) {
            next_call->month = 1;
            ++next_call->year;
        }
        next_call->day -= days_per_month[next_call->month] -
20      1; // Roll over to proper day.
    }
    // Adjust the day back to BCD.
    if (LoNibble( next_call->day ) > 0x9 || HiNibble(
25  next_call->day ) != HiNibble( old_day ))
        next_call->day += 6;

    // Adjust the month to BCD.
    if (LoNibble( next_call->month ) > 0x9) next_call->
30  month += 6;

    // Adjust the year back to BCD.
    if (LoNibble( next_call->year ) > 0x9) next_call->year
    += 6;
    if (HiNibble( next_call->year ) > 0x9) next_call->year
35  = LoNibble( next_call->year );
    }
    */

40  #define INCL_DOSNMPPIPES
    #include <os2.h>

    #include <iostream.h>
    #include <fstream.h>
    #include <string.h>
45  #include <server.h>

    #include "DBServer.H"

50  #include <usertype.h>
    #include <DB_Objects.HPP>
    #include <CTID.H>
    #include <CTIMS.HPP>
    #include <CTMessage.HPP>
55  #include <MessagePipe.HPP>
```

SUBSTITUTE SHEET

```
FLAG fProcessClientEvent( MessagePipe &Pipe, TStream
&MsgStream );

5  FLAG fProcessQueryCTIDStatus( MessagePipe &Pipe,
    QueryCTIDStatusMsg &Status );
    FLAG fProcessStoreMonitorEvent( MessagePipe &Pipe,
    StoreMonitorEventMsg &MEvent );
    FLAG fUpdateLicenseStatus( StoreMonitorEventMsg& );

10  // Helper functions.
    FLAG _fCopyTstoDBVars( char *tsstring, short *indicator,
    CTimestamp &ts, STRING varname = "Timestamp" );

    DataBase DB;

15  int main( int argc, char *argv[] )
    {
        if (argc != 3) {
            cout << "Usage: dbserver <database_name>
20  <pipe_name>" << endl;
        }

        DB.SetName( argv[1] );
        SvrMsgPipeFactory Factory( argv[2], 512, 10 );
25  MessagePipe *pipe;

        if (!DB.fConnect()) {
            cout << "Unable to connect to " << argv[1] << "
30  SQLCODE = " << (long)DB.ulSQLCode() << endl;
            return 1;
        }

        if (!Factory.fCreatePipe( pipe )) {
            cout << "Unable to create pipe DosErrorCode = " <<
35  Factory.rcDosErrorCode() << endl;
            return 2;
        }

        cout << "Waiting for pipe to connect to client..." <<
endl;
40  if (!pipe->fOpenPipe()) {
            cout << "Error connecting to the client
            DosErrorCode = " << pipe->rcDosErrorCode() << endl;
            return 2;
        }

45  cout << "Pipe connected to client." << endl;

        TStream MsgStream;
        while (fProcessClientEvent( *pipe, MsgStream ))
            MsgStream.Reset();
50  pipe->fClosePipe();
        return 0;
    }
}
```

SUBSTITUTE SHEET

```
FLAG fProcessClientEvent( MessagePipe &Pipe, TStream
&MsgStream )
{
5   if (!Pipe.fGetMessage( MsgStream )) {
       cout << "Error reading message from pipe
DosErrorCode = " << Pipe.rcDosErrorCode() << endl;
       return FALSE;
   }

10   CNetMessageHeader Header;
   MsgStream >> Header;
   switch (Header.eType()) {
       case QUERY_CTID_STATUS:
15       {
           QueryCTIDStatusMsg StatusMsg( Header );
           MsgStream >> *(QueryCTIDStatus*)&StatusMsg;
           if (!fProcessQueryCTIDStatus( Pipe, StatusMsg ))
               cout << "Error in fProcessQueryCTIDStatus,
20   SQLCODE = " << (long)ulGetSQLCode() << endl;
           }
           break;
           case STORE_MONITOREVENT:
25       {
               StoreMonitorEventMsg EventMsg( Header );
               MsgStream >> *(StoreMonitorEvent*)&EventMsg;
               if (!fProcessStoreMonitorEvent( Pipe, EventMsg
30   )) {
                   cout << "Error in fProcessStoreMonitorEvent,
                   SQLCODE = " << (long)ulGetSQLCode() << endl;
               }
               break;
               case CLI_QUIT:
35               return FALSE;
               default:
                   cout << "Unknown Command Received!" << endl;
                   return FALSE;
               }
40   return TRUE;
}

FLAG fProcessQueryCTIDStatus( MessagePipe &Pipe,
QueryCTIDStatusMsg &CTID )
45 {
    CTlicense Rec;
    CTIDStatusResultMsg ResultMsg;

    if (!fxlatCliCTID( CTID.CTID, CTID.CTID )) {
50   cout << "Error converting client CTID to server
CTID" << endl;
        // Proccess error here.
    }
}
```

SUBSTITUTE SHEET

```
ResultMsg.QueryResult = _fQueryLicense( &Rec,
CTID.CTID );

5      if (!ResultMsg.QueryResult) {
        ResultMsg.CTID = CTID.CTID;
        ResultMsg.Status =
CTLicStatus::ACTIVE;
        ResultMsg.PeriodDays = 2;
10      ResultMsg.PeriodMinutes = 0;
        ResultMsg.StolenFlag = FALSE;
        ResultMsg.SpecialProcess = 0;
        ResultMsg.Orgnum_n .fSetNull();
        ResultMsg.LastCallTS_n .fSetNull();
        ResultMsg.NextCallTS_n .fSetNull();
15      ResultMsg.NextCallClientTS_n .fSetNull();
        ResultMsg.ProductType .fSetNull();
      }
      else {
20      ResultMsg.CTID = Rec.CTID;
        ResultMsg.Status = Rec.LicStatus;
        ResultMsg.PeriodDays = Rec.PeriodDays;
        ResultMsg.PeriodMinutes = Rec.PeriodMinutes;
        ResultMsg.StolenFlag = Rec.StolenFlag ==
'Y';
25      ResultMsg.SpecialProcess = Rec.SpecialProcess;
        ResultMsg.LastCallTS_n .Assign(
Rec.LastCallTS_N, DB_ISNULL( Rec.IsNull_LastCallTS ) );
        ResultMsg.NextCallTS_n .Assign(
30      Rec.NextCallTS_N, DB_ISNULL( Rec.IsNull_NextCallTS ) );
        ResultMsg.NextCallClientTS_n .Assign(
Rec.NextCallClientTS_N, DB_ISNULL(
Rec.IsNull_NextCallClientTS ) );
        if (DB_ISNULL( Rec.IsNull_Orgnum ))
35      ResultMsg.Orgnum_n .fSetNull();
        else
        ResultMsg.Orgnum_n = Rec.Orgnum_N;
        ResultMsg.ProductType = Rec.ProductType;
      }

40      cout << "SQLCODE = " << (long)ulGetSQLCode() << endl;

      // Return Query results.
      TStream Stream;
      Stream << ResultMsg;
45      return Pipe.fSendMessage( Stream );
    }

50      FLAG fProcessStoreMonitorEvent( MessagePipe &Pipe,
StoreMonitorEventMsg &Msg )
    {
      StoreResultMsg ResultMsg;

      // Prepare reply message.
55      ResultMsg.Result = TRUE;
```

SUBSTITUTE SHEET

```
// Prepare the monitorevent data.
    _CTmonitorEvent Rec;

    if (!fXlatCliCTID( (ULONG&)Rec.CTID, Msg.CTID )) {
5      cout << "Error converting client CTID to server
      CTID" << endl;
      // Process error here.
    }

10    _fCopyTStoDBVars( Rec.ServerTS, NULL,
    Msg.ServerTS, "ServerTS" );
    _fCopyTStoDBVars( Rec.ClientTS, NULL,
    Msg.ClientTS, "ClientTS" );
    _fCopyTStoDBVars( Rec.TelcoTS_N, &Rec.IsNull_TelcoTS,
15    Msg.TelcoTS_n, "TelcoTS" );

    Rec.DurationSec_N = Msg.DurationSec_n;
    Rec.IsNull_DurationSec = DB_NOT_NULL;

20    if (!Msg.CallerID_n) {
        Rec.IsNull_CallerID = DB_NULL;
    }
    else {
        Rec.IsNull_CallerID = DB_NOT_NULL;
25        strncpy( Rec.CallerID_N, Msg.CallerID_n, sizeof(
        Rec.CallerID_N ) );
    }

    Rec.LineNum = Msg.LineNum;

30    if (!Msg.LogFlag) {
        cout << "INVALID DATA ERROR: LogFlag is NULL,
        defaulting to FALSE" << endl;
        Rec.LogFlag = 'N';
35    }
    else {
        Rec.LogFlag = ((STRING)Msg.LogFlag)[0];
    }

40    strncpy( Rec.EnvironmentID, Msg.EnvironmentID, sizeof(
    Rec.EnvironmentID ) );

    Rec.ErrorCnt = Msg.ErrorCnt;

45    // Update the License Record.
    if (!fUpdateLicenseStatus( Msg )) {
        if (ulGetSQLCode() != 100) {
            cout << "DB2_ERROR: Error updating License
            Table, CliCTID = " << Msg.CTID
50            << " SQLCODE = " << (long)ulGetSQLCode() <<
            endl;
        }
    }

55    // Perform the insert.
```

SUBSTITUTE SHEET

```

    if (!_fInsertIntoMonitorEvent( &Rec )) {
        ResultMsg.Result = FALSE;
    }
    else {
5       if (Msg.StoreAsStolen) {
            if (!_fInsertIntoMonitorEventStolen( &Rec )) {
                ResultMsg.Result = FALSE;
            }
        }
10      if (Msg.StoreAsExpire) {
            if (!_fInsertIntoMonitorEventExpired( &Rec )) {
                ResultMsg.Result = FALSE;
            }
        }
15    }

    cout << "SQLCODE = " << (long)ulGetSQLCode() << endl;

    TStream Stream;
20    Stream << ResultMsg;
    if (Pipe.fSendMessage( Stream ) && ResultMsg.Result ==
TRUE) {
        DB.Commit();
        return TRUE;
25    }
    else {
        DB.Rollback();
        return FALSE;
    }
30 }

FLAG fUpdateLicenseStatus( StoreMonitorEventMsg &Msg )
{
35    CTupdateLicenseStatus Rec;
    short dummy1; // Used to quiet the
    Null validation below.

    fxlatCliCTID( (ULONG&)Rec.CTID, Msg.CTID );
40    strncpy( Rec.Status, Msg.LicenseStatus, sizeof(
Rec.Status ) );

    _fCopyTStoDBVars( Rec.LastCallTS_N, &dummy1,
Msg_ServerTS, "LastCallTS" );
45    _fCopyTStoDBVars( Rec.NextCallTS_N, &dummy1,
Msg_NextCallTS_n, "NextCallTS" );
    _fCopyTStoDBVars( Rec.NextCallClientTS_N, &dummy1,
Msg_NextCallClientTS_n, "NextCallClientTS" );

50    if (!Msg.NextCallTS_n) strcpy( Rec.NextCallTS_N,
"0001-01-01-00.00.00.000000" );
    if (!Msg.NextCallClientTS_n) strcpy(
Rec.NextCallClientTS_N, "0001-01-01-00.00.00.000000" );

55    return _fUpdateLicenseStatus( &Rec );

```

SUBSTITUTE SHEET


```

}

5  FLAG_fCopyTstoDBVars( char *tsstring, short *indicator,
  CTTimestamp &ts, STRING varname )
{
    if (!ts) {
        if (indicator == NULL) {
            cout << "INVALID_DATA_ERROR: " << varname << "
10  is NULL, forcing validation" << endl;
            ts.ForceValidate();
        }
        else {
            *indicator = DB_NULL;
15  tsstring[0] = '\x0';
            return FALSE;
        }
    }
    else if (!ts.fValidate()) {
20  cout << "INVALID_DATA_ERROR: " << varname << " is
    invalid, forcing validation" << ts << endl;
    ts.ForceValidate();
    }
25  if (indicator != NULL) *indicator = DB_NOT_NULL;
    ts.ToSTRING( tsstring );
    return TRUE;
}

30

#define INCL_NOPMAPI          // no PM in this program
35  #define INCL_DOS
    #define INCL_BSE
    #define INCL_DOSSEMAPHORES
    #define INCL_DOSNMPIPES
    #include <os2.h>

40  #include <ctype.h>
    #include <stdlib.h>
    #include <iostream.h>
    #include <fstream.h>

45  #include <server.h>

    #include <MessagePipe.HPP>
    #include <TModem.HPP>

50  #include "CT_Trans.H"

/*GLOBAL
VARIABLES*****/
55  HEV hQuitSem;
```

SUBSTITUTE SHEET

```
// Temp, move to thread.
CltMsgPipeFactory *factory;
MessagePipe *pipe;

5  /*****
   **/

FLAG fLoadLineThreads( TModem&, PCSZ, PCSZ );
void _Optlink CT_CommandThread( PVOID );
10 FLAG fParseCmd( TPort &Port, TConnectInfo *CnctInfo,
   STRING buffer );

TPort::ComSettings ComSetting = {
15   "COM1",          // port name
   0,               // not used
   38400,           // bps
   8,              // data bits
   TPort::NO,       // no parity
20   TPort::ONE       // one stop bit
};

int main( int argc, char *argv[] )
{
25   APIRET rc;

   cout << "CompuTrace Server V0.99q" << endl;

   // Check arguments.
   if (argc != 4) {
30       cout << "Usage: server <pipe_name> <port_name>
<init_string>" << endl << endl;
       return 0;
   }

35   // Create quit semaphore.
   if ((rc = DosCreateEventSem( NULL, &hQuitSem, 0, FALSE
   )) != 0)
       return 1;

40   factory = new CltMsgPipeFactory( argv[1], 512 );

   // Load port server threads.
   TPort Port;
   TModem Modem = Port;
45   if (!fLoadLineThreads( Modem, argv[2], argv[3] ))
       return 2;

   cout << "Successfully connected to local modem" <<
50   endl;

   // Wait for quit signal.
   DosWaitEventSem( hQuitSem, SEM_INDEFINITE_WAIT );

   return 0;
55 }
```

SUBSTITUTE SHEET

```
5 //
  // fLoadLineThreads: Loads the threads to operate a
  // server line. This function
  // should be called for each server
  // line.
  //
  FLAG fLoadLineThreads( TModem &Modem, PCSZ port_str, PCSZ
10 init_str )
  {
    // Start port log.
    // Port.LogOn();

    // Open port.
15 ComSetting.port_name = port_str;
    if (!Modem.Port().fOpenPort( ComSetting )) {
      cout << "Error openning port" << endl;
      return FALSE;
    }

    // Start the port manage thread.
    if (!Modem.Port().fStartManageThread()) {
      cout << "Thread execution error" << endl;
      return FALSE;
25 }

    // Initialize the modem.
    STRING result = Modem.strSendCommand( init_str, -1 );
    if (strcmp( result, "OK" ) != 0) {
      cout << "Error initiallizing modem" << endl;
      return FALSE;
30 }

    // Connect pipe to dbserver.
    if (!factory->fCreatePipe( pipe )) return FALSE;
    if (!pipe->fOpenPipe()) return FALSE;

    // Start the command thread.
    if (!Modem.Port().fStartCommandThread(
40 CT_CommandThread, (PVOID)&Modem )) {
      cout << "Thread execution error" << endl;
      Modem.Port().KillManageThread();
      return FALSE;
    }

    return TRUE;
  }

50 //
  // CT_CommandThread: Processes incoming data from a
  // server line.
  //
  void _Optlink CT_CommandThread( PVOID ptr )
55 {
```

SUBSTITUTE SHEET

```
TModem &Modem = *(TModem*)ptr;           // Alias
(should be optimized out by the compiler).

// Thread local variables
5  STRING result;
   TConnectInfo cnct_info;

   while (TRUE) {
       result = Modem.strGetString( -1 );
10  // Parse buffer for cmd.
       if (!fParseCmd( Modem.Port(), &cnct_info, result ))
       {
           memset( (PVOID)&cnct_info, '\x0', sizeof
15  cnct_info );
       }
   }

#define CND_DATE_FIELD      "DATE ="
20 #define CND_TIME_FIELD   "TIME ="
#define CND_NUMBER_FIELD    "NMBR ="

#define CND_NONUM_FIELD     "REASON FOR NO NUMBER:"
#define CND_NAME_FIELD      "CALLER NAME:"
25 #define CND_NONAME_FIELD  "REASON FOR NO NAME:"

//
// fParseCmd: called when a '\n' has been received, this
// function will process the string.
// Returns TRUE if a transaction is occuring,
30 FALSE if the buffers should be cleared.
//

FLAG fParseCmd( TPort &Port, TConnectInfo *cnct_info,
35 STRING buffer )
{
    const char *index;

    // Parse command.
    if (strstr( buffer, "RING" ) != NULL) {
40         cout << "Command parsed as RING" << endl;
    }
    else if ((index = strstr( buffer, CND_DATE_FIELD )) !=
50  NULL) {
        index += sizeof CND_DATE_FIELD;
        while (!isdigit( *index )) index++;
        // Grab the month.
        if (!isdigit( *index ) || !isdigit( *(index+1) ))
        return FALSE;
        cnct_info->cnd.month = (*index++ - '0') * 10;
        cnct_info->cnd.month += *index++ - '0';
        // Grab the day.
        if (!isdigit( *index ) || !isdigit( *(index+1) ))
        return FALSE;
55         cnct_info->cnd.day = (*index++ - '0') * 10;
```

SUBSTITUTE SHEET

```

    cnct_info->cnd.day += *index++ - '0';
    cout << buffer << endl;
}
5   else if ((index = strstr( buffer, CND_TIME_FIELD )) !=
    NULL) {
        index += sizeof CND_TIME_FIELD;
        while (!isdigit( *index )) index++;
        // Grab the hour.
10    if (!isdigit( *index ) || !isdigit( *(index+1) ))
        return FALSE;
        cnct_info->cnd.hour = (*index++ - '0') * 10;
        cnct_info->cnd.hour += *index++ - '0';
        // Grab the minute.
15    if (!isdigit( *index ) || !isdigit( *(index+1) ))
        return FALSE;
        cnct_info->cnd.minute = (*index++ - '0') * 10;
        cnct_info->cnd.minute += *index++ - '0';
20    cout << buffer << endl;
    }
    else if ((index = strstr( buffer, CND_NUMBER_FIELD ))
    != NULL) {
        index += sizeof CND_NUMBER_FIELD;
25    while (isspace( *index )) index++;
        // Grab the number.
        for (int i = 0; i < CND_NUM_MAXLEN; i++) {
            if (index[i] == '\x0' || index[i] == '\r') {
30                cnct_info->cnd.number[i] = '\x0';
                break;
            }
            else {
                cnct_info->cnd.number[i] = index[i];
35            }
        }
        cout << buffer << endl;
    }
    else if (strstr( buffer, CND_NONUM_FIELD ) != NULL) {
40    index += sizeof CND_NONUM_FIELD;
        // Grab the string.
        while (isspace( *index )) index++;
        for (int i = 0; i < CND_NUM_MAXLEN; i++) {
            if (index[i] == '\x0' || index[i] == '\r') {
45                cnct_info->cnd.number[i] = '\x0';
                break;
            }
            else {
                cnct_info->cnd.number[i] = index[i];
50            }
        }
        cout << buffer << endl;
    }
    else if (strstr( buffer, CND_NAME_FIELD ) != NULL) {
55    index += sizeof CND_NAME_FIELD;
```

SUBSTITUTE SHEET

```

// Grab the name.
while (isspace( *index )) index++;
for (int i = 0; i < CND_NAME_MAXLEN; i++) {
5   if (index[i] == '\x0' || Index[i] == '\r') {
        cnct_info->cnd.name[i] = '\x0';
        break;
    }
    else {
10        cnct_info->cnd.name[i] = index[i];
    }
}

cout << buffer << endl;
}
15 else if (strstr( buffer, CND_NONAME_FIELD ) != NULL)
{
    index += sizeof CND_NONAME_FIELD;
    // Grab the string.
    while (isspace( *index )) index++;
20    for (int i = 0; i < CND_NAME_MAXLEN; i++) {
        if (index[i] == '\x0' || Index[i] == '\r') {
            cnct_info->cnd.name[i] = '\x0';
            break;
        }
        else {
25            cnct_info->cnd.name[i] = index[i];
        }
    }

    cout << buffer << endl;
}
30 else if (strstr( buffer, "CONNECT" ) != NULL) {
    cout << "Command parsed as CONNECT" << endl;

35    SntlConnect( Port, *pipe, cnct_info );
    return FALSE;
}
else if (strstr( buffer, "NO CARRIER" ) != NULL) {
40    cout << "Command parsed as NO CARRIER" << endl;
    return FALSE;
}
else if (strstr( buffer, "OK" ) != NULL) {
    cout << "Command parsed as OK" << endl;
45    return FALSE;
}
else if (strstr( buffer, "ERROR" ) != NULL) {
    cout << "Command parsed as ERROR" << endl;
    return FALSE;
}
50 else {
    cout << "Unknown command received: " << buffer <<
endl;
    return FALSE;
}
55 return TRUE;

```

SUBSTITUTE SHEET

```

    }

#include <CTIMS.HPP>

5 //=====
//
// CTStatus friends and members.
//
10 CTStatus::CTStatus()
{
    memset( value, ' ', sizeof( value ) );
}

15 CTStatus::CTStatus( STRING str )
{
    ASSERT( strlen( str ) < sizeof( value ) );
    memcpy( value, str, strlen( str ) );
20
}

const char CTLicStatus::STR_SET[][CT_TOK_SIZE+1] = {
25     UNUSED_TOK,
    NOTEST_TOK,
    ACTIVE_TOK,
    EXPIRED_TOK
};

30
CTLicStatus& CTLicStatus::operator = ( STRING str )
{
    for (int i = 0; i <= EXPIRED; i++) {
35         if (strcmp( STR_SET[i], str ) == NULL) {
            setNotNull();
            value = VALUE( i );
            return *this;
        }
    }
40     ASSERT( FALSE ); // No match was found
    for the string.
    return *this;
}

45 /*****
FLAG CTOrgnum::fSetPrefix( STRING str )
{
    if (strlen( str ) != ORGNUM_PREFIX_SIZE) {
50         return FALSE;
    }
    else {
        value[0] = str[0];
        value[1] = str[1];
        value[2] = str[2];
55         value[3] = str[3];
    }
}
```

SUBSTITUTE SHEET

```
        return TRUE;
    }
}

5  FLAG CTOrgnum::fSetIndex( UINT num )
{
    if (num > 9999) {
        return FALSE;
    }
10  else {
        value[ORGNUM_PREFIX_SIZE + 0] = (num%10000) / 1000
        + '0';
        value[ORGNUM_PREFIX_SIZE + 1] = (num%1000) / 100 +
        '0';
15  value[ORGNUM_PREFIX_SIZE + 2] = (num%100) / 10 +
        '0';
        value[ORGNUM_PREFIX_SIZE + 3] = (num % 10) + '0';
    }
}

20  FLAG CTOrgnum::fGetPrefix( char *str ) const
{
    if (strlen( str ) != ORGNUM_PREFIX_SIZE) {
25  return FALSE;
    }
    else {
        str[0] = value[0];
        str[1] = value[1];
        str[2] = value[2];
30  str[3] = value[3];
        str[4] = '\x0';
    }
}

35  FLAG CTOrgnum::fGetIndex( UINT &i ) const
{
    i = atoi( &(value[ORGNUM_PREFIX_SIZE]) );
    return TRUE;
}

40  FLAG CTOrgnum::fGeneratePrefix( STRING org_name )
{
    char pre[ORGNUM_PREFIX_SIZE];

45  // Grab first four alphanum characters.
    for (int i = 0, j = 0; i < ORGNUM_PREFIX_SIZE;) {
        if (isalnum( orgname[j++] )) pre[i];
    }
}

50  *****/

//*****
//*****
//
55  // iostream stream operators.
```

SUBSTITUTE SHEET

- 139 -

```

//
ostream& operator <<( ostream &os, const CTStatus &status
)
{
5   return os << (STRING)status;
}

//*****
//*****
10 //
// TStream stream operators.
//
TStream& operator << ( TStream &buf, const CTStatus
&status )
15 {
    buf << *(TNull*)&status;
    if (!status) return buf;
    else return buf.Put( PVOID( status.value ), sizeof(
20 status.value ) );
}

TStream& operator >> ( TStream &buf, CTStatus &status )
{
25     buf >> *(TNull*)&status;
    if (!status) return buf;
    else return buf.Get( status.value, sizeof(
status.value ) );
}

30 TStream& operator << ( TStream &buf, const CTCallerID &id
)
{
    buf << *(TNull*)&id;
    if (!id) return buf;
35     else return buf.Put( PVOID( id.value ), sizeof(
id.value ) );
}

40 TStream& operator >> ( TStream &buf, CTCallerID &id )
{
    buf >> *(TNull*)&id;
    if (!id) return buf;
    else return buf.Get( id.value, sizeof( id.value ) );
45 }

TStream& operator << ( TStream &buf, const CTLicStatus
&lic )
{
50     buf << *(TNull*)&lic;
    if (!lic) return buf;
    else return buf << USHORT( lic.value );
}

55 TStream& operator >> ( TStream &buf, CTLicStatus &lic )
{

```

SUBSTITUTE SHEET

- 140 -

```

    USHORT num;

    buf >> *(TNull*)&lic;
    if (!lic) return buf;
5      else {
        buf >> num;
        lic.value = CTLicStatus::VALUE( num );
        return buf;
      }
10    }

TStream& operator << ( TStream &buf, const CTOrgnum &num
)
{
15    buf << *(TNull*)&num;
    if (!num) return buf;
    else return buf.Put( PVOID( num.value ), sizeof(
num.value ) );
}
20

TStream& operator >> ( TStream &buf, CTOrgnum &num )
{
    buf >> *(TNull*)&num;
    if (!num) return buf;
25    else return buf.Get( num.value, sizeof( num.value ) );
}

TStream& operator << ( TStream &buf, const CTMonitorEvent
&event )
30    {
        return buf << event.CTID
            << event.ServerTS
            << event.ClientTS
35            << event.TelcoTS_n
            << event.DurationSec_n
            << event.CallerID_n
            << event.LineNum
            << event.LogFlag
40            << event.EnvironmentID
            << event.ErrorCnt;
    }

TStream& operator >> ( TStream &buf, CTMonitorEvent
&event )
45    {
        return buf >> event.CTID
            >> event.ServerTS
            >> event.ClientTS
50            >> event.TelcoTS_n
            >> event.DurationSec_n
            >> event.CallerID_n
            >> event.LineNum
            >> event.LogFlag
55            >> event.EnvironmentID

```

SUBSTITUTE SHEET

- 141 -

```

        >> event.ErrorCnt;
    }

5      #include <CTMessage.HPP>

      //*****
      //*****
10     //
      // TStream stream operators.
      //
      TStream& operator << ( TStream &buf, const
15     CTMessageHeader &head )
      {
          return buf << head.ID << head.Type << head.Len;
      }

      TStream& operator >> ( TStream &buf, CTMessageHeader
20     &head )
      {
          buf >> head.ID;
          buf >> head.Type;
          buf >> head.Len;
25     return buf;
      }

      #define INCL_NOPMAPI                // no PM in this program
30     #define INCL_DOS
      #define INCL_BSE
      #define INCL_DOSSEMAPHORES
      #define INCL_DOSNMPIPES
      #include <os2.h>
35     #include "CT_Buffer.HPP"

      CT_Buffer::CT_Buffer()
          : head( 0 ),
          tail( CT_BUFFER_MAXLEN )
      {
          // Create the mutex sem.
          rc = DosCreateMutexSem( NULL, &hBufSem, 0, 0 );
          if (rc) {}
45     // Create the event sem.
          rc = DosCreateEventSem( NULL, &hReleaseGetSem, 0, 0 );
      }

50     CT_Buffer::~CT_Buffer()
      {
          DosCloseMutexSem( hBufSem );
      }

55     void CT_Buffer::Flush()

```

SUBSTITUTE SHEET

- 142 -

```

    {
        ULONG post_count;

        DosRequestMutexSem( hBufSem, SEM_INDEFINITE_WAIT );
5       head = 0;
        tail = CT_BUFFER_MAXLEN;
        DosResetEventSem( hReleaseGetSem, &post_count );
        DosReleaseMutexSem( hBufSem );
    }
10
    FLAG CT_Buffer::fPutChar( char ch )
    {
        FLAG ret_val;

15       // Get ownership of the semaphore.
        rc = DosRequestMutexSem( hBufSem, SEM_INDEFINITE_WAIT );
        if (rc) return FALSE;

20       // First check that the log buffer hasn't overflowed.
        if (!fIsFull()) {
            // Store the char, update head, signal the event.
            buffer[head] = ch;
            head = IncBufPtr( head );
25         DosPostEventSem( hReleaseGetSem );
            ret_val = TRUE;
        }
        else ret_val = FALSE;

30       // Release the semaphore.
        DosReleaseMutexSem( hBufSem );

        return ret_val;
    }
35
    FLAG CT_Buffer::fGetChar( char &ch )
    {
        ULONG post_count;
        FLAG ret_val;

40       // If empty wait for timeout.
        if (fIsEmpty()) DosWaitEventSem( hReleaseGetSem,
            SEM_INDEFINITE_WAIT );

45       // Get ownership of the semaphore.
        rc = DosRequestMutexSem( hBufSem, SEM_INDEFINITE_WAIT );
        if (rc) return FALSE;

50       if (!fIsEmpty()) {
            // Fetch the char, update tail.
            tail = IncBufPtr( tail );
            ch = buffer[tail];
            ret_val = TRUE;
55       }
    }

```

CIRCTITITE SHEET

- 143 -

```

        else ret_val = FALSE;

        DosResetEventSem( hReleaseGetSem, &post_count );

5      // Release the semaphore.
        DosReleaseMutexSem( hBufSem );

        return ret_val;
    }

10

#define INCL_NOPMAPI                // no PM in this program
#define INCL_DOS
#define INCL_BSE
15  #define INCL_DOSSEMAPHORES
#define INCL_DOSNMPPIPES
#include <os2.h>

#include "CT_Log.HPP"

20  #include <fstream.h>

CT_Log::CT_Log( UINT len )
    :   buf_len( len ),
        index( 0 )
{
    if ((buffer = new BYTE[buf_len]) == NULL) {
        buf_len = index = 0;
    }
30 }

CT_Log::~CT_Log()
{
    if (buffer) DosFreeMem( buffer );
35 }

BOOL CT_Log::fPostChar( char ch )
{
    // First check that the log buffer hasn't overflowed.
40  if (!fIsFull()) {
        // Store the char, update head.
        buffer[index++] = ch;
        return TRUE;
    }
45  else return FALSE;
}

BOOL CT_Log::fDumpLog( const char *fname )
{
50  fstream dump;

    dump.open( fname, ios::out );
    if (!dump) return FALSE;
    dump.write( buffer, index );
55  dump.close();

```

SUBSTITUTE SHEET

- 144 -

```

        return TRUE;
    }

    #define INCL_DOSNMPPIPES
    #include <os2.h>

    #include <MessagePipe.HPP>

    //*****
    // SvrMsgPipeFactory Implementation.
    //*****

    SvrMsgPipeFactory::SvrMsgPipeFactory( PCSZ name, UINT
msg_len, UINT pipe_len )
        : MsgPipeFactory( msg_len ),
          pipe_name( name ),
          pipe_len( pipe_len )
    {}

    FLAG SvrMsgPipeFactory::fCreatePipe( MessagePipe *ppipe
)
    {
        ppipe = new MessagePipe( this );

        return TRUE;
    }

    FLAG SvrMsgPipeFactory::fDestroyPipe( MessagePipe *ppipe
)
    {
        delete ppipe;

        return TRUE;
    }

    FLAG SvrMsgPipeFactory::fOpenPipe( MessagePipe *pipe )
    {
        HPIPE hPipe;

        // Create and connect the named pipe.
        pipe->rc = DosCreateNPipe( (PSZ)pipe_name, &hPipe,
NP_NOWRITEBEHIND |
Data sent to remote pipes immediatly.
NP_ACCESS_DUPLEX,
Two-way client/server communications.
NP_WAIT |
I/O to pipe blocked until data available.
NP_TYPE_MESSAGE |
Message pipe type.
NP_READMODE_MESSAGE |
Messafe read mode type.
0x00FF,
Infinite number of allowed instances of this pipe.

```

- 145 -

```

        (uMaxMsgLen() + 2) * pipe_len, //
Size of output buffer.
        (uMaxMsgLen() + 2) * pipe_len, //
Size of input buffer.
5      0 //
Client open timeout (see DosWaitNPipe).
        );
        if (pipe->rc) return FALSE;
10      pipe->rc = DosConnectNPipe( hPipe );
        if (pipe->rc) return FALSE;

        pipe->SetHandle( hPipe );
        return TRUE;
15    }

FLAG SvrMsgPipeFactory::fClosePipe( MessagePipe *pipe )
{
    HPIPE hPipe = pipe->GetHandle();
20
    // Wait till the pipe is empty.
    pipe->rc = DosResetBuffer( hPipe );
    if (pipe->rc) return FALSE;
    // Disconnect the pipe handle.
25    pipe->rc = DosDisconnectNPipe( hPipe );
    if (pipe->rc) return FALSE;

    return TRUE;
30    }

//*****
//*****
// CltMsgPipeFactory Implementation.
//*****
35 //*****

ClMsgPipeFactory::ClMsgPipeFactory( PCSZ name, UINT
msg_len )
    : MsgPipeFactory( msg_len ),
40    pipe_name( name )
{}

FLAG CltMsgPipeFactory::fCreatePipe( MessagePipe *&ppipe
)
45 {
    ppipe = new MessagePipe( this );

    return TRUE;
50 }

FLAG CltMsgPipeFactory::fDestroyPipe( MessagePipe *ppipe
)
{
55    delete ppipe;

```

SUBSTITUTE SHEET

- 146 -

```

        return TRUE;
    }

    FLAG CltMsgPipeFactory::fOpenPipe( MessagePipe *pipe )
5   {
        HPIPE hPipe;
        ULONG ulAction;

        pipe->rc = DosOpen( pipe_name, &hPipe, &ulAction, 0,
10         FILE_NORMAL, FILE_OPEN,
            OPEN_ACCESS_READWRITE |
            OPEN_SHARE_DENYNONE,
            (PEAOP2) NULL );
        if (pipe->rc) return FALSE;
15     pipe->SetHandle( hPipe );
        return TRUE;
    }

20   FLAG CltMsgPipeFactory::fClosePipe( MessagePipe *pipe )
    {
        HPIPE hPipe = pipe->GetHandle();

        // Wait till the pipe is empty.
25     pipe->rc = DosResetBuffer( hPipe );
        if (pipe->rc) return FALSE;
        // Close the pipe handle.
        rc = DosClose( hPipe );
        if (pipe->rc) return FALSE;
30     return TRUE;
    }

35     //*****
    // MessagePipe Implementation
    //*****

40   MessagePipe::MessagePipe( MsgPipeFactory *mom )
        : factory( mom )
    {
        factory->InitPipe( this );
    }

45   MessagePipe::~MessagePipe()
    {
        factory->DeinitPipe( this );
    }

50   FLAG MessagePipe::fOpenPipe()
    {
        return factory->fOpenPipe( this );
    }

55

```

INSTITUTE SHEET

- 147 -

```

FLAG MessagePipe::fClosePipe()
{
    return factory->fClosePipe( this );
}

5 FLAG MessagePipe::fSendMessage( PCVOID msg, ULONG msg_len
)
{
    ULONG cbWritten;
10    rc = DosWrite( hPipe, (PVOID)msg, msg_len, &cbWritten
);

    return (rc == 0 && msg_len == cbWritten) ? TRUE :
15 FALSE;
}

FLAG MessagePipe::fGetMessage( PVOID msg, PULONG msg_len
)
20 {
    // PRECONDITION( msg_len != 0 && *msg_len <=
uMaxMsgLen() );

    rc = DosRead( hPipe, msg, *msg_len, msg_len );
25    return (rc == 0) ? TRUE : FALSE;
}

FLAG MessagePipe::fTransact( PCVOID out_msg, ULONG
out_msg_len, PVOID in_msg, PULONG in_msg_len )
30 {
    // PRECONDITION( in_msg_len != 0 && *in_msg_len <=
uMaxMsgLen() );

    rc = DosTransactNPipe( hPipe, (PVOID)out_msg,
out_msg_len, in_msg, *in_msg_len, in_msg_len );
35    return (rc == 0) ? TRUE : FALSE;
}

40 MessagePipe::PIPE_STATE MessagePipe::eState()
{
    ULONG cbRead;
    AVAILDATA avail;
45    ULONG state;

    // Use DosPeekNPipe to find the state of the pipe.
    rc = DosPeekNPipe( hPipe, NULL, 0, &cbRead, &avail,
&state );
50    return (PIPE_STATE)state;
}

#ifdef OS2
55 #define INCL_DOSDATETIME

```

SUBSTITUTE SHEET

- 148 -

```

#include <os2.h>
#endif

#include <ctype.h>
5
#include <Objects.HPP>

//*****
//
10 // TFlag members.
//

TFlag::TFlag()
15 : TNull( TRUE )
{}

TFlag::TFlag( FLAG flag )
: value( (flag != FALSE) ),
20 TNull( FALSE )
{}

TFlag::~TFlag()
{
25 #ifdef DEBUG
    fSetNull();
    value = UNINIT_DATA;
#endif
}

//*****
//
30 // TTimestamp members.
//

const UINT TTimestamp::TSStringLen = 27;

TTimestamp::TTimestamp()
40 : TNull( TRUE )
{
    #ifdef DEBUG
        Year = Month = Day = Hour = Minute = Second =
45 Millisec = UNINIT_DATA;
    #endif
}

TTimestamp::TTimestamp( USHORT yr, UCHAR mo, UCHAR dy,
50 USHORT ms )
    UCHAR hr, UCHAR mn, UCHAR sc,
: Year( yr ),
  Month( mo ),
  Day( dy ),
  Hour( hr ),
55 Minute( mn ),

```

SUBSTITUTE SHEET

- 149 -

```

        Second( sc ),
        Millisec( ms ),
        TNull( FALSE )
    {}
5
    TTimestamp::~TTimestamp()
    {
        #ifdef DEBUG
            fSetNull();
10        Year = Month = Day = Hour = Minute = Second =
        Millisec = UNINIT_DATA;
        #endif
    }

15    FLAG TTimestamp::fValidate() const
    {
        if (fIsNull()) return FALSE;

        // Check year.
20        if (!Year || Year > 9999) return FALSE;
        // Check month and day.
        if (!Day) return FALSE;
        switch (Month) {
25            case 1:
                if (Day > 31) return FALSE;
                break;
            case 2:
                if (Year % 4 == 0 && Year % 100 != 0) //
30                Check for a leapyear.
                if (Day > 29) return FALSE;
                else
                    if (Day > 28) return FALSE;
                    break;
            case 3:
35                if (Day > 31) return FALSE;
                break;
            case 4:
                if (Day > 30) return FALSE;
                break;
40            case 5:
                if (Day > 31) return FALSE;
                break;
            case 6:
                if (Day > 30) return FALSE;
45                break;
            case 7:
                if (Day > 31) return FALSE;
                break;
            case 8:
50                if (Day > 31) return FALSE;
                break;
            case 9:
                if (Day > 30) return FALSE;
                break;
55            case 10:

```

- 150 -

```

        if (Day > 31) return FALSE;
        break;
    case 11:
        if (Day > 30) return FALSE;
        break;
    case 12:
        if (Day > 31) return FALSE;
        break;
    default:
        return FALSE;
    }
    // Check hours.
    if (Hour > 23) {
        if (Hour > 24 || Minute || Second || Millisec)
            return FALSE;
    }
    // Check minutes, seconds and milliseconds.
    if (Minute > 59 || Second > 59 || Millisec > 999)
        return FALSE;

    return TRUE;
}

void TTimestamp::ForceValidate()
{
    setNotNull();
    Year = Month = Day = 1;
    Hour = Minute = Second = Millisec = 0;
}

FLAG TTimestamp::IsValidTSString( STRING ts )
{
    if (
        isdigit( ts[0] )           // Check Year.
        && isdigit( ts[1] )
        && isdigit( ts[2] )
        && isdigit( ts[3] )
        && ts[4] == '-'
        && isdigit( ts[5] )       // Check Month.
        && isdigit( ts[6] )
        && ts[7] == '-'
        && isdigit( ts[8] )       // Check Day.
        && isdigit( ts[9] )
        && ts[10] == '-'
        && isdigit( ts[11] )      // Check Hour.
        && isdigit( ts[12] )
        && ts[13] == '.'
        && isdigit( ts[14] )     // Check Minute.
        && isdigit( ts[15] )
        && ts[16] == '.'
        && isdigit( ts[17] )     // Check Second.
        && isdigit( ts[18] )
        && ts[19] == '.'
        && isdigit( ts[20] )     // Check Millisec.
        && isdigit( ts[21] )
        && isdigit( ts[22] )
    )

```

SUBSTITUTE SHEET

- 151 -

```

        && isdigit( ts[23] )
        && isdigit( ts[24] )
        && isdigit( ts[25] )
        && ts[26] == '\x0' )
5      return TRUE;
      else return FALSE;
    }

10  TTimestamp& TTimestamp::Assign( const TTimestamp &ts )
    {
        if (!ts) {
            fSetNull();
        }
        else {
15          setNotNull();
            Year = ts.Year;
            Month = ts.Month;
            Day = ts.Day;
            Hour = ts.Hour;
20          Minute = ts.Minute;
            Second = ts.Second;
            Millisec = ts.Millisec;
        }
        return (*this);
25    }

    TTimestamp& TTimestamp::Assign( USHORT yr, UCHAR mo,
    UCHAR dy,
                                UCHAR hr, UCHAR mn, UCHAR
30    sc, USHORT ms )
    {
        setNotNull();

        Year = yr;
35        Month = mo;
        Day = dy;
        Hour = hr;
        Minute = mn;
        Second = sc;
40        Millisec = ms;

        return (*this);
    }

45  TTimestamp& TTimestamp::Assign( STRING ts, FLAG isnull )
    {
        unsigned num;

        if (isnull) {
50          fSetNull();
            return *this;
        }

        setNotNull();
55    }

```

SUBSTITUTE SHEET

- 152 -

```

    ASSERT( fIsValidTSString( ts ) );

    /* Convert year */
    num = (ts[0] - '0') * 1000;
    num += (ts[1] - '0') * 100;
    num += (ts[2] - '0') * 10;
    num += (ts[3] - '0');
    Year = USHORT( num );
    /* Convert month */
    num = (ts[5] - '0') * 10;
    num += (ts[6] - '0');
    Month = UCHAR( num );
    /* Convert day */
    num = (ts[8] - '0') * 10;
    num += (ts[9] - '0');
    Day = UCHAR( num );
    /* Convert hour */
    num = (ts[11] - '0') * 10;
    num += (ts[12] - '0');
    Hour = UCHAR( num );
    /* Convert minute */
    num = (ts[14] - '0') * 10;
    num += (ts[15] - '0');
    Minute = UCHAR( num );
    /* Convert second */
    num = (ts[17] - '0') * 10;
    num += (ts[18] - '0');
    Second = UCHAR( num );
    /* Convert millisec */
    num = (ts[20] - '0') * 100;
    num += (ts[21] - '0') * 10;
    num += (ts[22] - '0');
    Millisec = USHORT( num );

    return *this;
}

#ifdef __OS2__
TTimeStamp& TTimeStamp::Assign( const DATETIME &Date )
{
    setNotNull();

    Year = Date.year;
    Month = Date.month;
    Day = Date.day;
    Hour = Date.hours;
    Minute = Date.minutes;
    Second = Date.seconds;
    Millisec = Date.hundredths * 10;

    return (*this);
}
#endif // __OS2__

STRING TTimeStamp::ToString( char *ts ) const

```

- 153 -

```

    {
        unsigned num;

        /* Convert year */
5       num = Year;
        ts[0] = (num%10000) / 1000 + '0';
        ts[1] = (num%1000) / 100 + '0';
        ts[2] = (num%100) / 10 + '0';
        ts[3] = (num % 10) + '0';
10      ts[4] = '-';
        /* Convert month */
        num = Month;
        ts[5] = (num%100) / 10 + '0';
        ts[6] = (num % 10) + '0';
15      ts[7] = '-';
        /* Convert day */
        num = Day;
        ts[8] = (num%100) / 10 + '0';
        ts[9] = (num % 10) + '0';
20      ts[10] = '-';
        /* Convert hour */
        num = Hour;
        ts[11] = (num%100) / 10 + '0';
        ts[12] = (num % 10) + '0';
25      ts[13] = '.';
        /* Convert minute */
        num = Minute;
        ts[14] = (num%100) / 10 + '0';
        ts[15] = (num % 10) + '0';
30      ts[16] = '.';
        /* Convert second */
        num = Second;
        ts[17] = (num%100) / 10 + '0';
        ts[18] = (num % 10) + '0';
35      ts[19] = '.';
        /* Convert millisec */
        num = Millisec;
        ts[20] = (num%1000) / 100 + '0';
        ts[21] = (num%100) / 10 + '0';
40      ts[22] = (num % 10) + '0';
        ts[23] = '0';
        ts[24] = '0';
        ts[25] = '0';

45      ts[26] = '\\x0';

        return ts;
    }

50  FLAG TTimestamp::operator > ( const TTimestamp &ts )
    const
    {
        useAsValue();

55      if (Year > ts.Year) return TRUE;

```

SUBSTITUTE SHEET

- 154 -

```

else if (Year == ts.Year) {
    if (Month > ts.Month) return TRUE;
    else if (Month == ts.Month) {
        if (Day > ts.Day) return TRUE;
        else if (Day == ts.Day) {
            if (Hour > ts.Hour) return TRUE;
            else if (Hour == ts.Hour) {
                if (Minute > ts.Minute) return TRUE;
                else if (Minute == ts.Minute) {
                    if (Second > ts.Second) return TRUE;
                    else if (Second == ts.Second) {
                        if (Millisec > ts.Millisec) return
15 TRUE;
                        else return FALSE;
                    }
                }
            }
        }
    }
}
20 }
    }
    return FALSE;
}

25 FLAG TTimestamp::operator >= ( const TTimestamp &ts )
const
{
    return (*this > ts || *this == ts);
}

30 FLAG TTimestamp::operator == ( const TTimestamp &ts )
const
{
    useAsValue();
35     if (Year == ts.Year &&
        Month == ts.Month &&
        Day == ts.Day &&
        Hour == ts.Hour &&
        Minute == ts.Minute &&
        Second == ts.Second &&
        Millisec == ts.Millisec) {
40         return TRUE;
    }
    else {
45         return FALSE;
    }
}

// Date and time add function.
50 TTimestamp& TTimestamp::AddToDate( UINT yr, UINT mon,
    UINT day,
    UINT hr, UINT min,
    UINT sec, UINT ms )
{
55     if (!IsNull()) {

```

CIRCTITITE CLEET

- 155 -

```

        ms += Millisec;
        sec += Second;
        min += Minute;
        hr  += Hour;
5       day += Day;
        mon += Month;
        yr  += Year;
    }

10    // Adjust and carry ms.
        while (ms > usMaxMillisec()) {
            ms -= usMaxMillisec() + 1;
            sec++;
        }
15    // Adjust and carry sec.
        while (sec > usMaxSecond()) {
            sec -= usMaxSecond() + 1;
            min++;
        }
20    // Adjust and carry min.
        while (min > usMaxMinute()) {
            min -= usMaxMinute() + 1;
            hr++;
        }
25    // Adjust and carry hr.
        while (hr > usMaxHour()) {
            hr -= usMaxHour() + 1;
            day++;
        }
30    // Adjust and carry mon (day adjust is dependent on mon
    and yr).
        while (mon > usMaxMonth()) {
            mon -= usMaxMonth();
            yr++;
35    }
    // Now adjust and carry day now that yr and mon is known.
        while (day > usMaxDay( yr, mon )) {
            day -= usMaxDay( yr, mon );
            mon++;
40            if (mon > usMaxMonth()) {
                mon -= usMaxMonth();
                yr++;
            }
        }
45
    // Copy new values to members.

        Assign( yr, mon, day, hr, min, sec, ms );

50    CHECK( fValidate() );
    return *this;
}

// static member.
55 USHORT TTimestamp::usMaxDay( USHORT year, USHORT month )

```

SUBSTITUTE SHEET

- 156 -

```

{
    switch (month) {
        case 1:          // Jan.
            return 31;

        case 2:          // Feb.
            return fIsLeapYear( year ) ? 29 : 28;

        case 3:          // Mar.
            return 31;

        case 4:          // Apr.
            return 30;

        case 5:          // May.
            return 31;

        case 6:          // Jun.
            return 30;

        case 7:          // Jul.
            return 31;

        case 8:          // Aug.
            return 31;

        case 9:          // Sep.
            return 30;

        case 10:         // Oct.
            return 31;

        case 11:         // Nov.
            return 30;

        case 12:         // Dec.
            return 31;

        // default:
        // BOILERPLATE;
    }

    //*****
    //
    // TStream stream operators.
    //
    TStream& operator << ( TStream &buf, const TFlag &flag )
    {
        if (!flag) return buf << FLAG( TRUE );
        else return buf << FLAG( FALSE ) << flag.value;
    }

    TStream& operator >> ( TStream &buf, TFlag &flag )

```

SUBSTITUTE SHEET

- 157 -

```

    {
        buf >> *(TNull*)&flag;
        if (flag.isNull() == FALSE)
            buf >> flag.value;
5       return buf;
    }

TStream& operator << ( TStream &buf, const TTimestamp &ts
10      )
    {
        if (!ts) return buf << FLAG( TRUE );
        else {
            return buf << FLAG( FALSE )
15                  << ts.Year
                  << ts.Month
                  << ts.Day
                  << ts.Hour
                  << ts.Minute
20                  << ts.Second
                  << ts.Millisec;
        }
    }

TStream& operator >> ( TStream &buf, TTimestamp &ts )
25      {
        buf >> *(TNull*)&ts;
        if (!ts) {
            return buf;
        }
        else {
30            return buf >> ts.Year
                        >> ts.Month
                        >> ts.Day
                        >> ts.Hour
35                        >> ts.Minute
                        >> ts.Second
                        >> ts.Millisec;
        }
    }
40

//*****
//*****
//
// iostream friend function members.
45 //

ostream& operator << ( ostream &os, const TFlag &flag )
    {
        if (!flag) return os << NULL_TOK;
50        else return os << (STRING)flag;
    }

/*****
istream& operator << ( istream &is, TFlag &flag )
55 {

```

RESTITUTE SHEET

- 158 -

```

    char ch, buffer[12];

    is >> ws;                                     // Extract leading
    whitespace.

5      for (int i = 0; i < sizeof( buffer ); i++) {
          is >> buffer[i];
          if (!isalpha( buffer[i] )) break;
      }
10     if (i == sizeof( buffer ) ASSERT( FALSE );

        buffer[i] = '\x0';

    if (strcmp( buffer, NULL_TOK) == 0) {
15         fSetNull();
    }
    else if (strcmp( buffer, TRUE_TOK) == 0) {
        Assign( TRUE );
    }
20     else if (strcmp( buffer, FALSE_TOK) == 0) {
        Assign( FALSE );
    }
    else ASSERT( FALSE );

25     return is;
}
    *****/

30     ostream& operator << ( ostream &os, const TTimestamp &ts
    {
        char tsstring[TTimestamp::TSStringLen];
        if (!ts) return os << "NULL";
        else return os << ts.ToSTRING( tsstring );
35     }

#define INCL_NOPMAPI                               // no PM in this program
40     #define INCL_DOS
        // #define INCL_BSE
        // #define INCL_DOSSEMAPHORES
        #include <os2.h>

        #include <usertype.h>
45     #include <TModem.HPP>

    TModem::TModem( TPort &_port )
        : port( _port )
    {}

50     TModem::RC TModem::rcSendCommand( STRING, ULONG timeout )
    {
        NOTIMPLEMENTED;
    }

55

```

SUBSTITUTE SHEET

- 159 -

```

STRING TModem::strSendCommand( STRING str, ULONG timeout
)
{
    port.fWritePort( str );
    port.fPutChar( '\r' );
5    STRING result = strGetString( timeout );
    if (strcmp( str, result ) == 0) {
        return strGetString( timeout );
    }
10    else {
        return result;
    }
}

15    STRING TModem::strGetString( ULONG timeout )
    {
        UINT i = 0;
        last_result[0] = '\x0';

20        // Eat Leading CR/NL.
        while (!port.fGetChar( last_result[i] )
            || last_result[i] == '\r'
            || last_result[i] == '\n') {}
        i++;
        // (already got 1 char ok)
25        // Grab text until a CR/NL.
        while (port.fGetChar( last_result[i] )
            && last_result[i] != '\n'
            && last_result[i] != '\r'
            && i <= sizeof( last_result )) {
30            i++;
        }
        last_result[i] = '\x0'; // Null terminate
        buffer.
        return last_result;
35    }

#include <TObject.HPP>

//*****
40 //*****
//
// TObject members.
//

45 TObject::~TObject()
{}

//*****
50 //*****
//
// TNull members.
//

TNull::TNull( FLAG is_null )
55 : isnull( is_null )

```

SIIRSTITUTE SHEET

- 160 -

```

    {}

    FLAG TNull::fSetNull()
    {
5       isnull = TRUE;
        return TRUE;
    }

10
    #define INCL_NOPMAPI
    #define INCL_DOS
    #define INCL_BSE
    #define INCL_DOSSEMAPHORES
15    #define INCL_DOSNMPPIPES
    #include <os2.h>

    #include <usertype.h>
    #include "TPacket.HPP"
20

    TPacket::TPacket( TPort& p )
        : Port( p ),
          text_length( 0 ),
          state( TRANS_NULL )
25    {}

    TPacket::TRANS_STATE TPacket::rGetPacket()
    {
        enq_count = 0;
30        nak_count = 0;
        text_length = 0;

        if (state != TRANS_NULL) return TRANS_NULL;

35    // Enquiry Loop.
        while (fSendENQ())
        {
            if ((state = rReceivePacket()) == TRANS_NAK)
            {
40                while (fSendNAK())
                    if ((state = rReceivePacket()) == TRANS_ACK)
                    {
                        fSendACK();
                        return state;
                    }
            }

            else if (state == TRANS_ACK)
50            {
                fSendACK();
                return state;
            }
        }
55    fSendEOT();

```

SIIRSTITUTE SHEET

- 161 -

```

        return state;
    }

5   TPacket::TRANS_STATE TPacket::rReceivePacket()
    {
        char ch;
        int i=0,j;

10  // Get STX.
        if (!Port.fGetChar( ch ))
            return TRANS_ETX;
        // packet_text[i++] = ch;
        if (ch != STX)
15  // return TRANS_NAK;

        // Get Length.
        if (!Port.fGetChar( ch ))
            return TRANS_NAK;
20  // packet_text[i++] = ch;

        text_length = (USHORT)ch;

        if (!Port.fGetChar( ch ))
25  // return TRANS_NAK;
        // packet_text[i++] = ch;

        text_length = (USHORT)(ch << 8) + text_length;

30  if (text_length > MAX_TEXT_LEN)
        return TRANS_NAK;

        // Get Text.

35  for (j=0 ; j < text_length; j++ )
        {
            if ( Port.fGetChar( ch ))
                packet_text[ j ] = ch;

40  // else
            // return ( TRANS_NAK );
        }

        // Get ETX.
45  if ( Port.fGetChar( ch ))
        {
            if ( ch == ETX )
            {
                // packet_text[ i++ ] = ch;
50  // else
                // return ( TRANS_NAK );
            }
        }
        else
55  {

```

SIIRSTITIITE SHEET

- 162 -

```

        return ( TRANS_NAK );
    }

    // Get LRC.
5    if (!Port.fGetChar( ch ))
        return TRANS_NAK;
    // packet_text[i++] = ch;
    return TRANS_ACK;
}

10    UINT TPacket::cbCopyText( PVOID ptr, UINT len )
    {
        len = len < text_length ? len : text_length;
        memcpy( ptr, packet_text, len );
15        return len;
    }

    FLAG TPacket::fSendENQ()
    {
20        char enq = ENQ;

        enq_count++;
        if (enq_count > MAX_ENQ) return FALSE;

25        Port.FlushInputBuffer();
        return Port.fWritePort( &enq, 1 );
    }

    FLAG TPacket::fSendACK()
30    {
        char ack = ACK;
        Port.FlushInputBuffer();
        return Port.fWritePort( &ack, 1 );
    }

35    FLAG TPacket::fSendNAK()
    {
        char nak = NAK;

40        nak_count++;
        if (nak_count > MAX_NAK) return FALSE;

        Port.FlushInputBuffer();
        return Port.fWritePort( &nak, 1 );
45    }

    FLAG TPacket::fSendEOT()
    {
50        char eot = EOT;
        return Port.fWritePort( &eot, 1 );
    }

#define INCL_NOPMAPI // no PM in this program
55 #define INCL_DOS

```

SUBSTITUTE SHEET

- 163 -

```

#define INCL_BSE
#define INCL_DOSSEMAPHORES
#define INCL_DOSNMPPIPES
#define INCL_DOSDEVIOCTL
5  #include <os2.h>

#define _THREADS // This implementation is
multi-threaded.

10  #include <process.h>
#include <string.h>
#include <stdlib.h>

#include "TPort.HPP"
15  TPort::TPort()
    : manage_thread( -1 ),
      log_flag( FALSE )
    {}

20  TPort::~TPort()
    {
        while (manage_thread != -1) {
            KillManageThread();
            DosSleep( 1000 ); // Wait 1 second.
        }
    }

30  FLAG TPort::fOpenPort( const ComSettings &settings )
    {
        LINECONTROL lctl;
        DCBINFO dcb;
        ULONG ulAction;
        ULONG ulPio, ulDio;
35  ULONG cbTrans;

        // Open the port.
        rc = DosOpen( settings.port_name, &hPort, &ulAction,
0, 0, OPEN_ACTION_OPEN_IF_EXISTS,
40  OPEN_FLAGS_WRITE_THROUGH |
OPEN_ACCESS_READWRITE | OPEN_SHARE_DENYREADWRITE, NULL );
        if (rc) return FALSE;

        // Set the line speed.
45  ulPio = sizeof( settings.bps );
        rc = DosDevIOctl( hPort, IOCTL_ASYNC,
ASYNC_SETBAUDRATE, (PVOID)&settings.bps,
        ulPio, &ulPio, NULL, 0, NULL );
        if (rc) {
50  DosClose( hPort );
            return FALSE;
        }

        // Set the line characteristics.
55  lctl.bDataBits = settings.data_bits;

```

SUBSTITUTE SHEET

- 164 -

```

    lctl.bParity = (BYTE)settings.parity;
    lctl.bStopBits = (BYTE)settings.stop_bits;
    ulPio = sizeof lctl;
    rc = DosDevIOctl( hPort, IOCTL_ASYNC,
5  ASYNC_SETLINECTRL, &lctl, ulPio, &ulPio, NULL, 0, NULL );
    if (rc) {
        DosClose( hPort );
        return FALSE;
    }
10
    // Set the flow control.
    ulDio = sizeof dcb;
    rc = DosDevIOctl( hPort, IOCTL_ASYNC,
    ASYNC_GETDCBINFO, NULL, 0, NULL, &dcb, ulDio, &ulDio );
15  if (rc) {
        DosClose( hPort );
        return FALSE;
    }
    /*****
    *****/
20  dcb.usReadTimeout = 100;

    dcb.fbCtlHndShake = MODE_CTS_HANDSHAKE; // flags1 =
25  00001000

    dcb.fbFlowReplace &= 0x30; // flags2 =
    00??0000
    dcb.fbFlowReplace |= MODE_RTS_HANDSHAKE; // flags2 =
30  10??0000

    dcb.fbTimeout &= 0xF8; // flags3 =
    ???000
    dcb.fbTimeout |= MODE_WAIT_READ_TIMEOUT; // flags3 =
35  ???100
    /*****
    *****/
    dcb.usReadTimeout = 300;
    dcb.fbCtlHndShake = MODE_CTS_HANDSHAKE;
    dcb.fbFlowReplace = MODE_RTS_HANDSHAKE;
40  dcb.fbTimeout = MODE_NO_WRITE_TIMEOUT |
    MODE_WAIT_READ_TIMEOUT;

    rc = DosDevIOctl( hPort, IOCTL_ASYNC,
    ASYNC_SETDCBINFO, &dcb, ulPio, &ulPio, NULL, 0, NULL );
45  if (rc) {
        DosClose( hPort );
        return FALSE;
    }

50  fRaisedDTR();

    return TRUE;
}

55  FLAG TPort::fClosePort()

```

SUBSTITUTE SHEET

- 165 -

```

    {
        rc = DosClose( hPort );
        if (rc) return FALSE;
        else return TRUE;
5    }

    void TPort::FlushInputBuffer()
    {
        BYTE cmd;
10    by API.                // Scratch, Needed
        ULONG len;
        by API.                // Scratch, Needed

        rc = DosDevIOctl( hPort, IOCTL_GENERAL,
15    DEV_FLUSHINPUT, &cmd, sizeof( cmd ), &len,
        &cmd, sizeof( cmd ), &len );

        DosSleep(10);          // Timing Kludge - Give the
        Device Driver          // time to flush buffer before
20    resetting                // semaphore stuff.

        buffer.Flush();
    }
25

    void TPort::FlushOutputBuffer()
    {
        BYTE cmd;
        by API.                // Scratch, Needed
30    ULONG len;
        by API.                // Scratch, Needed

        rc = DosDevIOctl( hPort, IOCTL_GENERAL,
35    DEV_FLUSHOUTPUT, &cmd, sizeof( cmd ), &len,
        &cmd, sizeof( cmd ), &len );
    }

    FLAG TPort::fReadPort( PVOID buf, UINT &len )
    {
40    for (int i = 0; i < len; i++) {
        if (buffer.fIsEmpty()) {
            len = i;
            return TRUE;
        }
45    else buffer.fGetChar( ((char*)buf)[i] );
    }
    return TRUE;
}

50    FLAG TPort::fWritePort( PVOID buf, UINT len )
    {
        ULONG cbWritten;

        rc = DosWrite( hPort, buf, len, &cbWritten );
55    if (rc) return FALSE;

```

SUBSTITUTE SHEET

- 166 -

```

        else return TRUE;
    }

    FLAG TPort::fDropDTR()
5    {
        ULONG ulPio, ulDio;
        MODEMSTATUS ms;
        ULONG com_err;

10        ms.fbModemOn = 0;
        ms.fbModemOff = DTR_OFF;
        ulPio = sizeof ms;
        ulDio = sizeof com_err;
        rc = DosDevIOctl( hPort, IOCTL_ASYNC,
15        ASYNC_SETMODEMCTRL, &ms, ulPio, &ulPio, &com_err, ulDio,
        &ulDio );
        if (rc) return FALSE;
        else return TRUE;
    }

20    FLAG TPort::fRaisedDTR()
    {
        ULONG ulPio, ulDio;
        MODEMSTATUS ms;
25        ULONG com_err;

        ms.fbModemOn = DTR_ON;
        ms.fbModemOff = 0xFF;
        ulPio = sizeof ms;
        ulDio = sizeof com_err;
30        rc = DosDevIOctl( hPort, IOCTL_ASYNC,
        ASYNC_SETMODEMCTRL, &ms, ulPio, &ulPio, &com_err, ulDio,
        &ulDio );
        if (rc) return FALSE;
35        else return TRUE;
    }

    void _Optlink ManageThread( PVOID ); // Used internally
    by fStartManageThread().
40    void _Optlink ManageThread( PVOID ptr )
    {
        ((TPort*)ptr)->ManagePort();
    }

45    FLAG TPort::fStartManageThread()
    {
        fManThread = TRUE;
        manage_thread = _beginthread( ManageThread, 8192,
50        (PVOID)this );
        if (manage_thread == -1) return FALSE;
        else return TRUE;
    }

55    void TPort::ManagePort()
    {

```

SUBSTITUTE SHEET

- 167 -

```

    char read_buf[32];
    ULONG cbRead;

    while (TRUE) {
5       rc = DosRead( hPort, read_buf, sizeof read_buf,
        &cbRead );
        if (rc) {
            // handle error here...
        }
        else if (!fManThread) break;
10      for (int i = 0; i < cbRead; i++) {
            if (log_flag) log.fPostChar( read_buf[i] );
            buffer.fPutChar( read_buf[i] );
        }
15      buffer.SignalRelease();
    }

    // Signal threads exit.
    manage_thread = -1;
20 }

FLAG TPort::fStartCommandThread( TTHREAD CommandThread,
PVOID data )
{
25     fCmdThread = TRUE;
    command_thread = _beginthread( CommandThread, 8192,
data );
    if (command_thread == -1) return FALSE;
    else return TRUE;
30 }

#include <TStream.HPP>

#include <debug.h>
35

#include <string.h>

//*****
//*****
40 //
// TStream members.
//
TStream::TStream( UINT buf_size )
:   buf_len( buf_size ),
45   buffer( new BYTE[buf_size] ),
    iptr( buffer ),
    xptr( buffer )
{
    #ifdef DEBUG
50     memset( buffer, UNDEF_DATA, buf_len );
    #endif
}

TStream::~TStream()
55 {

```

SUBSTITUTE SHEET

- 168 -

```

        delete buffer;
    }

    void TStream::Reset()
5    {
        iptr = xptr = buffer;
    }

    TStream& TStream::operator << ( const FLAG flag )
10    {
        *(FLAG*)iptr = flag;
        return incInserter( sizeof( flag ) );
    }

    TStream& TStream::operator << ( const USHORT num )
15    {
        *(USHORT*)iptr = num;
        return incInserter( sizeof( num ) );
    }

    TStream& TStream::operator << ( const ULONG num )
20    {
        *(ULONG*)iptr = num;
        return incInserter( sizeof( num ) );
    }

    TStream& TStream::operator << ( const char *str )
2    {
        strcpy( iptr, str );
30        return incInserter( strlen( str ) + 1 );
    }

    TStream& TStream::Put( const PVOID data, UINT size )
35    {
        memcpy( iptr, data, size );
        return incInserter( size );
    }

    TStream& TStream::operator >> ( FLAG &flag )
40    {
        flag = *(FLAG*)xptr;
        return incExtractor( sizeof( flag ) );
    }

    TStream& TStream::operator >> ( USHORT &num )
45    {
        num = *(USHORT*)xptr;
        return incExtractor( sizeof( num ) );
    }

    TStream& TStream::operator >> ( ULONG &num )
50    {
        num = *(ULONG*)xptr;
        return incExtractor( sizeof( num ) );
    }
55    }

```

SUBSTITUTE SHEET

- 169 -

```

TStream& TStream::operator >> ( char *str )
{
    strcpy( str, xptr );
    return incExtractor( strlen( str ) + 1 );
5   }

TStream& TStream::Get( PVOID data, UINT size )
{
10   memcpy( data, xptr, size );
    return incExtractor( size );
}

TStream& TStream::incExtractor( UINT n )
{
15   xptr += n;
    ASSERT( xptr <= iptr );
    return *this;
}

TStream& TStream::incInsertter( UINT n )
{
20   iptr += n;
    ASSERT( iptr <= buffer + buf_len );
    return *this;
25   }

;*****
;*****
30  ;*
;*   Copyright (C) 1995 Absolute Software Corporation
;*
;*****
;*****
35

NAME DBServer WINDOWCOMPAT

IMPORTS      CTIMS.fGenerateSerCTID
              CTIMS.fXlatSerCTID
40             CTIMS.fXlatCliCTID
              CTIMS.fGenerateCTCODE
              CTIMS.fConvertStrToCTCODE
              CTIMS.fConvertCTCODEToStr

45  .\TObject.obj: \
      f:\Server\TObject.CPP \
      DBServer.MAK

50  .\objects.obj: \
      f:\Server\objects.cpp \
      DBServer.MAK

55  .\MessagePipe.obj: \
      f:\Server\MessagePipe.CPP \
      DBServer.MAK

```

INSTITUTE SHEET

- 170 -

```

    .\CTMessage.obj: \
      f:\Server\CTMessage.CPP \
      DBServer.MAK
5    .\ctims.obj: \
      f:\Server\ctims.cpp \
      DBServer.MAK

10   .\DBServer.obj: \
      f:\Server\DBServer.C \

      {f:\Server;F:\Server\INCLUDE;E:\SQLLIB;E:\TOOLKT21\CPLUS\
      OS2H;E:\Tools\IBMCPP\INCLUDE;}DBServer.H \
      DBServer.MAK
15   .\TSTREAM.obj: \
      f:\Server\TSTREAM.CPP \
      DBServer.MAK

20   .\TPacket.obj: \
      f:\Server\TPacket.CPP \

      {f:\Server;M:\SRC\Include;M:\CT\Include;$(INCLUDE);}TPack
25   et.HPP \
      Server.MAK

      .\TModem.obj: \
      f:\Server\TModem.CPP \
      Server.MAK
30   .\CT_Log.obj: \
      f:\Server\CT_Log.CPP \

      {f:\Server;M:\SRC\Include;M:\CT\Include;$(INCLUDE);}CT_Lo
35   g.HPP \
      Server.MAK

      .\CT_Buffer.obj: \
      f:\Server\CT_Buffer.CPP \
40   .

      {f:\Server;M:\SRC\Include;M:\CT\Include;$(INCLUDE);}CT_Bu
      ffer.HPP \

      {f:\Server;M:\SRC\Include;M:\CT\Include;$(INCLUDE);}serve
45   r.h \
      Server.MAK

      .\Server.obj: \
      f:\Server\Server.C \
50   .

      {f:\Server;M:\SRC\Include;M:\CT\Include;$(INCLUDE);}CT_Tr
      ans.H \

      {f:\Server;M:\SRC\Include;M:\CT\Include;$(INCLUDE);}TPack
55   et.HPP \

```

SUBSTITUTE SHEET

- 171 -

```

    Server.MAK

    .\CT_Trans.obj: \
        f:\Server\CT_Trans.C \
5
    {f:\Server;M:\SRC\Include;M:\CT\Include;$(INCLUDE);}CT_Tr
    ans.H \

    {f:\Server;M:\SRC\Include;M:\CT\Include;$(INCLUDE);}TPack
10
    et.HPP \
        Server.MAK

    .\TPort.obj: \
        f:\Server\TPort.CPP \
15
    {f:\Server;M:\SRC\Include;M:\CT\Include;$(INCLUDE);}TPort
    .HPP \

    {f:\Server;M:\SRC\Include;M:\CT\Include;$(INCLUDE);}CT_Bu
20
    ffer.HPP \

    {f:\Server;M:\SRC\Include;M:\CT\Include;$(INCLUDE);}CT_Lo
    g.HPP \

    {f:\Server;M:\SRC\Include;M:\CT\Include;$(INCLUDE);}serve
25
    r.h \
        Server.MAK

    #ifndef CT_TRANS_H
30
    #define CT_TRANS_H

    // #include <DB_Objects.HPP>
    #include <MessagePipe.HPP>

    #include "TPacket.HPP"
35

    void SntlConnect( TPort &Port, MessagePipe &Pipe,
    TConnectInfo *cnct_info );
    void SntlDisconnect( TPort &Port, TConnectInfo
40
    &ConnectInfo );
    void SendDatePacket( TPort &Port, const SNTL_DATE &date
    );

    void AddDays( SNTL_DATE *next_call, int days );
45
    FLAG fGetDateTime( PDATETIME );

    #endif
    #ifndef MESSAGE_H
50
    #define MESSAGE_H

    /*****
    *****/
    Message.H
55

```

SUBSTITUTE SHEET

- 172 -

Defines all valid messages used by the Server and ServerShell.

```

5  *****
   *****/

   // Define standard types.
   #include <os2def.h>

10  #include <time.h>

   // Definition for the Sentinel date packet.
   struct CT_DATE {
15     BYTE year;
       BYTE month;
       BYTE day;
       BYTE hour;
       BYTE minute;
20   };

   // Definition for the Sentinel serial number packet.
   struct CT_SN {
       USHORT sn[3];
       USHORT cksum;
25   CT_DATE date;
   };

   #define CND_NUM_MAXLEN      20
   #define CND_NAME_MAXLEN    20
30

   struct CALLERID_INFO {
       BYTE month;
       BYTE day;
       BYTE hour;
35   BYTE minute;
       CHAR number[CND_NUM_MAXLEN];
       CHAR name[CND_NAME_MAXLEN];
   };

40   enum TRANS_STATE {
       TRANS_OK      = 0x00,
       TRANS_BAD_CND = 0x01,
       TRANS_BAD_SN  = 0x02,
       TRANS_BAD_DATE = 0x04
45   };

   struct CT_Transaction {
       DATETIME start_time;
       CALLERID_INFO cnd;
50   CT_SN sn;
       TRANS_STATE state;
       DATETIME end_time;
   };

55   enum CT_SN_QUERY {

```

CONFIDENTIAL SHEET

- 173 -

```

    CT_SN_OK           = 0,
    CT_SN_REDFLAG      = 1,
    CT_SN_UNKNOWN      = 2
};

5

#define CT_BUFFER_LEN 256 // Allowable
10 length of modem communications for a cycle.
#define CT_GUARD_CHAR    '!'

/* Definitions for stripped CompuTrace messages.
15 *****/

#define MAX_PHONE_NUM_LEN      16 // Max length of a
phone number string.
#define CT_SERIAL_NUM_LEN      sizeof( CT_SN ) //
20 Length of serial number packet sent by the modem.
#define MAX_ERROR_STR_LEN     32 // Max length of
an error string.

enum CTMSG_TYPE {
25     CTMSG_UNDEF = 0,
    CTMSG_CONNECT,
    CTMSG_SERIAL_NUM,
    CTMSG_ERROR_LOG,
    CTMSG_DISCONNECT
30 };

struct CT_ConnectMsg {
    time_t connect_time;
    char phone_num[MAX_PHONE_NUM_LEN];
35 };

struct CT_SerialNumMsg {
    CT_SN serial_num;
40 };

struct CT_ErrorLogMsg {
    char error_str[MAX_ERROR_STR_LEN];
45 };

struct CT_DisconnectMsg {
    time_t disconnect_time;
    char log[CT_BUFFER_LEN];
50 };

struct CTMessage {
    CTMSG_TYPE type;
    union {
        CT_ConnectMsg Connect;
        CT_SerialNumMsg SerialNum;
        CT_ErrorLogMsg ErrorLog;
55     CT_DisconnectMsg Disconnect;
    };
};

```

SUBSTITUTE SHEET

- 174 -

```

    } Msg;
};

#define MAX_CTMSG_SIZE sizeof( CTMessage )           // Max
5 size of a stripped (CompuTrace) message.

/* Definitions for pipe messages.
10 *****/

// Define all valid events. The following prefixes are
// used:
// CT_ For general messages
// CT_SER For server originated messages not
15 related to a transaction.
// CT_CLI For client originated messages not
related to a transaction.
// CT_SER_MSG For server originated messages related
to a transaction.
20 // CT_CLI_MSG For client originated messages related
to a transaction.
// For more detailed information please see the proper
message structure.
enum EVENT_TYPE {
25 CT_SER_MSG_AWK, // Server acknowledges
last received message.
CT_SER_MSG_ERROR, // Server has had a non-
fatal error.
CT_SER_MSG_FATAL, // Server has had a
30 fatal error and will unconditionally terminate.
CT_SER_MSG_MESSAGE, // Server has a message
to be processed by the client.

CT_SER_STOP, // Server requests the
35 client(s) stop sending messages.
CT_SER_START, // Server allows the
client(s) to continue sending messages.

CT_SER_ERROR, // Server has had an
40 internal non-fatal error.
CT_SER_FATAL, // Server has had an
internal fatal error and will terminate.
CT_SER_STRING, // Server has a general
string to be stored.
45 CT_SER_QUIT, // Server has requested
all clients to terminate.

CT_CLI_MSG_AWK, // Client acknowledges
50 last received message.
CT_CLI_MSG_ERROR, // Client has had a non-
fatal error.
CT_CLI_MSG_FATAL, // Client has had a
fatal error and will unconditionally terminate.
55 CT_CLI_MSG_MESSAGE // Client has a message
to be processed by the server.

```

C11DCTITITE CLEFT

- 175 -

```

};

// Define message transfer template used to transfer a
// message through a pipe.
5 struct CT_MessageHead {
    ULONG Id; // The message id
    number.
    EVENT_TYPE type; // The event type (see
    above).
10    BYTE len; // The length the
    message data.
};

struct CT_MessageBuffer {
15    CT_MessageHead header;
    char message[MAX_CTMSG_SIZE];
};

#define MAX_MSG_SIZE sizeof( CT_MessageBuffer )
20 // Max size of a pipe message.

#endif // MESSAGE_H

#ifdef PACKET_H
25 #define PACKET_H

// Ensure byte alignment enforced!
#pragma pack( 1 ) // For C-Set++
#pragma option -a1 // For BC++
30

/* Packet Level Defines
*****/
#define STX 0x02 // Start-of-
35 text.
#define ETX 0x03 // End-of-
text.
#define EOT 0x04 // End-of-
transmission.
#define ENQ 0x05 // Enquiry.
40 #define ACK 0x06 //
Acknowledgement.
#define NAK 0x15 // Negative-
acknowledgement.

45 #define MAX_ENQ 3 // Max
number of ENQs.
#define MAX_NAK 2 // Max
number of NAKs.

50 #define MAX_TEXT_LEN 256 // Max size
of a packets TEXT.

struct PKT_HEADER {
55    BYTE stx;
    BYTE lsb_length;

```

SUBSTITUTE SHEET

- 176 -

```

        BYTE msb_length;
    };

    struct PKT_FOOTER {
5        BYTE etx;
        BYTE lrc;
    };

10    /* Packet type definitions
    *****/

    // Text Type IDs.
    #define CTID_TEXT_TYPE (WORD) 0x0000 //
    Sentinel Subscription Number Packet.
15    #define NC_TEXT_TYPE (WORD) 0x0080 //
    Server Next Call Packet.

    struct SNTL_DATE {
20        BYTE year;
        BYTE month;
        BYTE day;
        BYTE hour;
        BYTE minute;
25    };

    struct CTID_TEXT {
        BYTE type;
        BYTE sub_type;
        WORD sn[3];
30    SNTL_DATE now_date;
    };

    #define SN_TEXT CTID_TEXT // Old name (uses
    should be changed to CTID_TEXT).

35    struct CTID_PACKET {
        PKT_HEADER header;
        CTID_TEXT text;
        PKT_FOOTER footer;
    };

40    #define SN_PACKET CTID_PACKET // Old name (uses
    should be changed to CTID_PACKET).

    struct NC_TEXT {
45        WORD type;
        SNTL_DATE next_call_date;
    };

    struct NC_PACKET {
50        PKT_HEADER header;
        NC_TEXT text;
        PKT_FOOTER footer;
    };

55    #pragma pack() // Back to default.
    #pragma option -a.

```

SUBSTITUTE SHEET

- 177 -

```

    #endif
    #ifndef SERVER_H
    #define SERVER_H

5      #define DEBUG          4

    #include <debug.h>
    #include <usertype.h>

10     //
    // TConnectInfo definition.
    //
    #define CND_NUM_MAXLEN      20
    #define CND_NAME_MAXLEN    20
15
    struct CALLERID_INFO {
        BYTE month;
        BYTE day;
        BYTE hour;
20        BYTE minute;
        CHAR number[CND_NUM_MAXLEN];
        CHAR name[CND_NAME_MAXLEN];
    };

25    struct TConnectInfo {
        DATETIME start_time, end_time;
        CALLERID_INFO cnd;
    };
    //
30    // End of TConnectInfo
    //

    #endif // SERVER_H
    #ifndef CT_BUFFER_HPP
35    #define CT_BUFFER_HPP

    #include "server.h"

    #define TRUE 1
    #define FALSE 0
40    #define CT_BUFFER_MAXLEN    256

    class CT_Buffer {

45        char buffer[CT_BUFFER_MAXLEN];
        UINT head, tail;
        HMTX hBufSem;
        HEV hReleaseGetSem;
50        APIRET rc;

        UINT IncBufPtr( UINT ptr ) const
            { return (++ptr >= CT_BUFFER_MAXLEN) ? 0 : ptr; }

55    public:

```

SUBSTITUTE SHEET

- 178 -

```

    CT_Buffer();
    ~CT_Buffer();

    void Flush();

5   BOOL fIsEmpty() const { return head == IncBufPtr( tail
    ); }
    BOOL fIsFull() const { return head == tail; }

10  void SignalRelease() { DosPostEventSem( hReleaseGetSem
    ); }

    BOOL fPutChar( char );
    BOOL fGetChar( char& );
15  };

    #endif
    #ifndef CT_LOG_HPP
    #define CT_LOG_HPP

20  #define TRUE 1
    #define FALSE 0

    class CT_Log {

25  char *buffer;
    UINT index, buf_len;

    public:

30  CT_Log( UINT = 4096 );
    ~CT_Log();

    void Flush() { index = 0; }

35  BOOL fIsEmpty() const { return index == 0; }
    BOOL fIsFull() const { return index >= buf_len; }

    BOOL fPostChar( char );

40  BOOL fDumpLog( const char * );
    };

    #endif
    #ifndef TCLIENT_HPP
    #define TCLIENT_HPP

50  class TClient {

    TConnectInfo ConnectInfo;
    WORD ctid[3];
55  SNTL_DATE client_date;

```

SUBSTITUTE CLIENT

Pipe

public:

5

}

10

15

```
#endif // CLIENT_HPP
#ifndef TPACKET_HPP
#define TPACKET_HPP
```

20

```
#include <os2def.h>
#include "packet.h"
```

25

```
#include <TPort.HPP>
```

```
/**
*****
```

30

```
// Class TPacket - Encapsulates the reception of a packet
for a port
```

```
//
```

```
// TPacket::TPacket( TPort& Port ) Initializes internal
state.
```

```
// Arguments:
```

35

```
// TPort& Port - the port to receive the packet
from.
```

```
//
```

```
// TRANS_STATE TPacket::rGetPacket()
```

```
// Description:
```

40

```
// Attempts to receive a packet from Port using the
protocol
```

```
// defined in the CompuTrace Protocol Specification
(CTPSpec).
```

```
//
```

```
// Returns: The result of the attempt:
```

45

```
// TRANS_ACK - packet successfully received as
defined by CTPSpec.
```

```
// TRANS_NAK - reception aborted due to invalid
reception, EOT sent.
```

50

```
// TRANS_ETO - ENQ timeout, no data recieved, EOT
sent.
```

```
//
```

```
// UINT TPacket::cbCopyText( ptr, len )
```

```
// Arguments:
```

55

```
// PVOID ptr - the buffer to copy data to.
```

```
// UINT len - the maximum number of bytes to copy.
```

SUBSTITUTE SHEET

```
//
//   Description:
//   Copies text from a sucessfully received packet
5   into buffer pointed to
//   by ptr. Copies up to len bytes or the size of
//   the received packet
//   text (whichever is smaller). Can only be called
//   if rGetPacket
//   returned TRANS_ACK.
10  //
//   Returns: number of bytes copied. or 0 if packet not
//   successfully
//   received.
//
15  // TRANS_STATE rState() const
//   Returns: the current state of the instance.
//*****
//*****
20  class TPacket {
public:

    enum TRANS_STATE {
        TRANS_NULL,
25  activity.
        TRANS_ACK,
        TRANS_NAK,
        TRANS_ETO };
    Enquiry time-out.
30  TPacket( TPort& );
    TRANS_STATE rGetPacket();
    UINT cbCopyText( PVOID ptr, UINT len );
35  TRANS_STATE rState() const { return state; }
protected:

    FLAG fSendENQ();
    FLAG fSendACK();
40  FLAG fSendNAK();
    FLAG fSendEOT();

private:
45  TPort& Port;
    int enq_count;
    int nak_count;
    USHORT text_length;
50  BYTE packet_text[MAX_TEXT_LEN];
    TRANS_STATE state;

    TRANS_STATE rReceivePacket();
};
55 #endif
```

SUBSTITUTE SHEET

```
# Created by IBM WorkFrame/2 MakeMake at 17:36:34 on
08/22/95
#
# This makefile should be run in the following directory:
5  # d:\Server
#
# The actions included in this makefile are:
#   COMPILE::CLC C++
#   LINK::CLC Link
10
.all: \
    .\DBServer.EXE

.SUFFIXES:

15 .SUFFIXES: .C .CPP

.CPP.obj:
    @echo WF::COMPILE::CLC C++
20    icc.exe /Tl- /Xi /ID:\Server\INCLUDE /IE:\SQLLIB
    /IE:\TOOLKT21\CPLUS\OS2H /IE:\Tools\IBMCPP\INCLUDE
    /DDEBUG=4 /Tdp /Q /Wall /Fi /Ti /Gm /G5 /Tm /C %s

.C.obj:
25    @echo WF::COMPILE::CLC C++
    icc.exe /Tl- /Xi /ID:\Server\INCLUDE /IE:\SQLLIB
    /IE:\TOOLKT21\CPLUS\OS2H /IE:\Tools\IBMCPP\INCLUDE
    /DDEBUG=4 /Tdp /Q /Wall /Fi /Ti /Gm /G5 /Tm /C %s

30 .\DBServer.EXE: \
    .\TObject.obj \
    .\TSTREAM.obj \
    .\DBServer.obj \
    .\ctims.obj \
35    .\CTMessage.obj \
    .\MessagePipe.obj \
    .\objects.obj \
    {$(LIB)}DB_Objects.LIB \
    {$(LIB)}SQL_DYN.LIB \
40    {$(LIB)}DBServer.DEF \
    DBServer.MAK
    @echo WF::LINK::CLC Link
    icc.exe @<<
45 /Tl- /Xi
    /ID:\Server\INCLUDE
    /IE:\SQLLIB
    /IE:\TOOLKT21\CPLUS\OS2H
    /IE:\Tools\IBMCPP\INCLUDE
50 /DDEBUG=4
    /Tdp /Q
    /Wall
    /Fi
    /Ti /Gm /G5 /Tm
    /B" /de"
55 /FeDBServer.EXE
```

SUBSTITUTE SHEET

```
DB_Objects.LIB
SQL_DYN.LIB
DBServer.DEF
5  .\TObject.obj
   .\TSTREAM.obj
   .\DBServer.obj
   .\ctims.obj
   .\CTMessage.obj
   .\MessagePipe.obj
10  .\objects.obj
   <<

!include DBServer.Dep
15 # Created by IBM WorkFrame/2 MakeMake at 10:20:11 on
   05/30/95
   #
   # This makefile should be run in the following directory:
20 #   d:\Server
   #
   # The actions included in this makefile are:
   #   COMPILE::CLC C++
   #   LINK::CLC Link

25 .all: \
   .\Server.EXE

.SUFFIXES:

30 .SUFFIXES: .C .CPP

.CPP.obj:
   @echo WF::COMPILE::CLC C++
   35   icc.exe /Tl- /ID:\Server\Include /IM:\CT\Include
   /Tdp /Q /Wall /Fi /Si /Ti /O /Gm /G5 /Tm /C %s

.C.obj:
   @echo WF::COMPILE::CLC C++
   40   icc.exe /Tl- /ID:\Server\Include /IM:\CT\Include
   /Tdp /Q /Wall /Fi /Si /Ti /O /Gm /G5 /Tm /C %s

.\Server.EXE: \
   .\TPacket.obj \
   .\TPort.obj \
45  .\CT_Trans.obj \
   .\Server.obj \
   .\CT_Buffer.obj \
   .\CT_Log.obj \
   .\TModem.obj \
50  {$(LIB)}CTIMS.LIB \
   {$(LIB)}MessagePipe.LIB \
   Server.MAK
   @echo WF::LINK::CLC Link
55  icc.exe @<<
/Tl-
```

SUBSTITUTE SHEET

```

/ID:\Server\Include
/IM:\CT\Include
/Tdp /Q
/Wall
5  /Fi /Si
   /Ti /O /Gm /G5 /Tm
   /B" /de"
   /FeServer.EXE
   CTIMS.LIB
10  MessagePipe.LIB
   .\TPacket.obj
   .\TPort.obj
   .\CT_Trans.obj
   .\Server.obj
15  .\CT_Buffer.obj
   .\CT_Log.obj
   .\TModem.obj
   <<

20  !include Server.Dep
   #define INCL_NOPMAPI           // no PM in this program.
   #define INCL_DOS
   #define INCL_BSE
25  #include <os2.h>
   #include <fstream.h>
   #include <time.h>

   #include <server.h>
30  #include <DB_Objects.HPP>
   #include <CTMessage.HPP>
   // #include <packet.h>
   #include "CT_Trans.H"

35  FLAG fQueryCTIDStatus( MessagePipe &Pipe, const
   QueryCTIDStatusMsg &Status, CTIDStatusResultMsg &Result
   );
   FLAG fStoreMonitorEvent( MessagePipe &Pipe, const
   StoreMonitorEventMsg &Store, StoreResultMsg &Result );
40  FLAG fSignalQuit( MessagePipe &Pipe );

   void AssignTS( TTimestamp &ts, const SNTL_DATE &Date );
   void AssignSNTL_DATE( SNTL_DATE &Date, const TTimestamp
   &ts );
45  // Temp function.
   void ProcessClient( TPort &Port, TConnectInfo
   &ConnectInfo, CTID_TEXT *text );

50  extern MessagePipe *pipe;

   //
   // SntlConnect: called when a CONNECT comand has been
   received, this function processes
```

SUBSTITUTE SHEET

```
//          a transaction between the server and a
// Sentinel client.
//
5 void SntlConnect( TPort &Port, MessagePipe &Pipe,
  TConnectInfo *cnct_info )
  {
    WORD msg_type;

    DosGetDateTime( &cnct_info->start_time );           //
10    Fill start time.

    TPacket packet( Port );

    while (TRUE) {
15      // Get a packet.
      if (packet.rGetPacket() != TPacket::TRANS_ACK) {
        cout << "Packet Error" << endl;
        return;
      }
20      // Determine packet type.
      packet.cbCopyText( &msg_type, sizeof( msg_type ) );
      switch( msg_type ) {
        case CTID_TEXT_TYPE:
          // Create a new client object.
25          TClient Client( Port, Pipe, *cnct_info );
          // Get CTID Text and add to Client object.
          CTID_TEXT Text;
          packet.cbCopyText( &Text, sizeof( Text ) );
          Client.SetCTID( Text );
30          // ProcessClient.
          ProcessClient( Client );
          ProcessClient( Port, *cnct_info, &Text );
          return;
        default:
35          return;
      }
    }
  }

40 void ProcessClient( TPort &Port, TConnectInfo
  &ConnectInfo, CTID_TEXT *text )
  {
    SNTL_DATE next_call;

45    // ENTER APPLICATION LAYER...

    // Query the Client state.
    QueryCTIDStatusMsg StatusMsg;
    StatusMsg.CTID = (ULONG)text->sn[0] + ((ULONG)text->
50    >sn[1] << 16);

    CTIDStatusResultMsg Result;

    cout << "QueryCTIDStatus for CTID " << StatusMsg.CTID
55    << "... ";
```

SUBSTITUTE SHEET

```
    if (!fQueryCTIDStatus( *pipe, StatusMsg, Result )) {
        cout << "Error in QueryCTIDStatus!" << endl;
    }
    else {
5       cout << "CTIDStatusResult Received..." << endl;
        cout << "    Status = " << (STRING)Result.Status <<
endl;      cout << "    PeriodDays = " << Result.PeriodDays <<
endl;      cout << "    PeriodMinutes = " <<
10      Result.PeriodMinutes << endl;
        cout << "    StolenFlag = " <<
        (STRING)Result.StolenFlag << endl;
        cout << "    SpecialProcess = " <<
15      Result.SpecialProcess << endl;
        cout << "    Orgnum = " << Result.Orgnum_n << endl;
    }

    // Send NextCall Message back to the Client.
    CTTimestamp next_ts;
    AssignTS( next_ts, text->now_date );
    if (next_ts.usYear() < 1900) { // If date is not
        valid substitute the local date instead.
        next_ts = ConnectInfo.start_time;
25    }
    next_ts.AddToDate( 0, 0, Result.PeriodDays, 0,
Result.PeriodMinutes );
    AssignSNTL_DATE( next_call, next_ts );

30    SendDatePacket( Port, next_call );
    SntlDisconnect( Port, ConnectInfo );

    // Store the Monitor Event.
    StoreMonitorEventMsg Event;
35    Event.StoreAsStolen = Result.StolenFlag;
    Event.StoreAsExpire = FALSE;

    Event.LicenseStatus = Result.Status;
    AssignTS( Event.ClientTS, text->now_date );
40    Event.ServerTS = ConnectInfo.start_time;
    Event.NextCallTS_n = Event.ServerTS;
    Event.NextCallTS_n.AddToDate( 0, 0, Result.PeriodDays,
0, Result.PeriodMinutes );
    Event.NextCallClientTS_n = next_ts;
45    Event.CTID = StatusMsg.CTID;
    Event.TelcoTS_n.Assign( Event.ServerTS.usYear(),
                            ConnectInfo.cnd.month,
                            ConnectInfo.cnd.day,
                            ConnectInfo.cnd.hour,
50                            ConnectInfo.cnd.minute );

    Event.DurationSec_n = 0;
    Event.CallerID_n = (const
char(*)[CALLERID_SIZE])ConnectInfo.cnd.number;
    Event.LineNum = 1;
55    Event.LogFlag = FALSE;
```

SUBSTITUTE SHEET

```
Event.EnvironmentID = "DBS-9508";
Event.ErrorCnt = 0;

5   StoreResultMsg ResultMsg;

   cout << endl << "Storing the MonitorEvent... ";

   if (!fStoreMonitorEvent( *pipe, Event, ResultMsg )) {
10      cout << "Error in StoreMonitorEvent!" << endl;
   }
   else {
      cout << "StoreResult = " << (ResultMsg.Result ?
15      "TRUE" : "FALSE") << endl;
   }
}

void SendDatePacket( TPort& Port, const SNTL_DATE& date )
{
20   NC_PACKET packet;

   packet.header.stx = STX;
   packet.header.lsb_length = sizeof( NC_TEXT );
25   packet.header.msb_length = 0;

   packet.text.type = NC_TEXT_TYPE;
   packet.text.next_call_date = date;

   packet.footer.etx = ETX;
30   packet.footer.lrc = 0;

   Port.fWritePort( (PVOID)&packet, sizeof( packet ) );
}

35

FLAG fQueryCTIDStatus( MessagePipe &Pipe, const
QueryCTIDStatusMsg &Status, CTIDStatusResultMsg &Result )
{
40   TStream in_strm, out_strm;

   out_strm << Status;
   if (!Pipe.fTransact( out_strm, in_strm )) return
FALSE;
   in_strm >> Result;
45

   if (Result.eType() == CTID_STATUS_RESULT) return TRUE;
   else return FALSE;
}

50
FLAG fStoreMonitorEvent( MessagePipe &Pipe, const
StoreMonitorEventMsg &Store, StoreResultMsg &Result )
{
   TStream in_strm, out_strm;

55   out_strm << Store;
```

SUBSTITUTE SHEET


```
    if (!Pipe.fTransact( out_strm, in_strm )) return
FALSE;
    in_strm >> Result;

5    if (Result.eType() == STORE_RESULT) return TRUE;
    else return FALSE;
}

10    FLAG fSignalQuit( MessagePipe &Pipe )
    {
        TStream stream;
        CliQuitMsg QuitMsg;

15        stream << QuitMsg;
        return Pipe.fSendMessage( stream );
    }

20    void SntlDisconnect( TPort &Port, TConnectInfo
    &ConnectInfo )
    {
        // Drop DTR.
        DosSleep( 500 );    // Broc - 13 Feb 95
25        // Add delay to let modem clear xmt
        buffer
        // to fix intermittent modem fault.
        Port.fDropDTR();

30        cout << "Disconnecting..." << flush;

        DosGetDateTime( &ConnectInfo.end_time );    //
        Fill end time.
        DosSleep( 200 );

35        // Raise DTR.
        Port.fRaisedDTR();
    }

40    // *** helper functions.
    UCHAR BCD2ToUChar( BYTE bcd )
    {
        // Convert a two digit bcd number to decimal.
45        return (bcd >> 4) * 10 + (bcd & 0x0F);
    }

    BYTE UCharToBCD2( UCHAR dec )
    {
50        // Convert a 8 bit decimal number to bcd.
        return (dec % 10) + (((dec / 10) % 10) << 4);
    }

    USHORT BCD4ToUShort( WORD bcd )
55    {
```

SUBSTITUTE SHEET

```

// Convert a four digit bcd number to decimal.
return (bcd >> 12) * 1000 + ((bcd & 0x0F00) >> 8) *
100 + ((bcd & 0x00F0) >> 4) * 10 + (bcd & 0x000F);
}

WORD UShortToBCD4( USHORT dec )
{
// Convert a 16 bit decimal number to a 4 digit decimal.
return (dec % 10) + (((dec / 10) % 10) << 4) + (((dec
/ 100) % 10) << 8) + (((dec / 1000) % 10) << 12);
}

void AssignTS( TTimestamp &ts, const SNTL_DATE &Date )
{
    ts.Assign( BCD2ToUChar( Date.year ),
               BCD2ToUChar( Date.month ),
               BCD2ToUChar( Date.day ),
               BCD2ToUChar( Date.hour ),
               BCD2ToUChar( Date.minute ) );
}

void AssignSNTL_DATE( SNTL_DATE &Date, const TTimestamp
&ts )
{
    Date.year   = UCharToBCD2( ts.usYear() % 100 );
    Date.month  = UCharToBCD2( ts.usMonth() );
    Date.day    = UCharToBCD2( ts.usDay() );
    Date.hour   = UCharToBCD2( ts.usHour() );
    Date.minute = UCharToBCD2( ts.usMinute() );
}

/*
inline BYTE HiNibble( BYTE b ) { return (BYTE)((b & 0xF0)
>> 4); }
inline BYTE LoNibble( BYTE b ) { return (BYTE)(b & 0x0F);
}

void AddDays( SNTL_DATE *next_call, int days )
{
    static BYTE days_per_month[18] = {
        0x31,
        0x28,
        0x30,          // 0x03 - March
        0x31,
        0x30,
        0x31,          // 0x06 - June
        0x30,
        0x31,
        0x30,          // 0x09 - Sept
        0x00,          // 0x0A
        0x00,          // 0x0B
        0x00,          // 0x0C
        0x00,          // 0x0D
        0x00,          // 0x0E
        0x00,          // 0x0F
    }
}

```

SUBSTITUTE SHEET

```

    0x31,      // 0x10 - Oct
    0x30,
    0x31      // 0x12 - Dec
};

5   BYTE old_day = next_call->day;
    // Save for BCD adjust.

    // Add the days to the current date.
10  next_call->day += days;
    // Check if we passed the end of the current month.
    if (next_call->day > days_per_month(next_call->month))
    {
        // Add one to month.
15      if (++next_call->month > 12) {
            next_call->month = 1;
            ++next_call->year;
        }
        next_call->day -= days_per_month(next_call->month) -
20      1; // Roll over to proper day.
    }
    // Adjust the day back to BCD.
    if (LoNibble( next_call->day ) > 0x9 || HiNibble(
25  next_call->day ) != HiNibble( old_day ))
        next_call->day += 6;

    // Adjust the month to BCD.
    if (LoNibble( next_call->month ) > 0x9) next_call->
30  >month += 6;

    // Adjust the year back to BCD.
    if (LoNibble( next_call->year ) > 0x9) next_call->year
    += 6;
    if (HiNibble( next_call->year ) > 0x9) next_call->year
35  = LoNibble( next_call->year );
    }
    */

#define INCL_DOSNMPPIPES
40  #include <os2.h>

    #include <iostream.h>
    #include <fstream.h>
    #include <string.h>
45  #include <server.h>

    #include "DBServer.H"

50  #include <usertype.h>
    #include <DB_Objects.HPP>
    #include <CTID.H>
    #include <CTIMS.HPP>
    #include <CTMessage.HPP>
55  #include <MessagePipe.HPP>
```

SUBSTITUTE SHEET

```
FLAG fProcessClientEvent( MessagePipe &Pipe, TStream
&MsgStream );

5 FLAG fProcessQueryCTIDStatus( MessagePipe &Pipe,
  QueryCTIDStatusMsg &Status );
FLAG fProcessStoreMonitorEvent( MessagePipe &Pipe,
StoreMonitorEventMsg &MEvent );
FLAG fUpdateLicenseStatus( StoreMonitorEventMsg& );

10 // Helper functions.
FLAG _fCopyTStoDBVars( char *tsstring, short *indicator,
CTTimestamp &ts, STRING varname = "Timestamp" );

15 DataBase DB;

int main( int argc, char *argv[] )
{
    if (argc != 3) {
20         cout << "Usage: dbserver <database_name>
<pipe_name>" << endl;
    }

    DB.SetName( argv[1] );
    SvrMsgPipeFactory( argv[2], 512, 10 );
25     MessagePipe *pipe;

    if (!DB.fConnect()) {
        cout << "Unable to connect to " << argv[1] << "
30     SQLCODE = " << (long)DB.ulSQLCode() << endl;
        return 1;
    }

    if (!Factory.fCreatePipe( pipe )) {
        cout << "Unable to create pipe DosErrorCode = " <<
35     Factory.rcDosErrorCode() << endl;
        return 2;
    }

    cout << "Waiting for pipe to connect to client..." <<
    endl;
40     if (!pipe->fOpenPipe()) {
        cout << "Error connecting to the client
        DosErrorCode = " << pipe->rcDosErrorCode() << endl;
        return 2;
    }

45     cout << "Pipe connected to client." << endl;

    TStream MsgStream;
    while (fProcessClientEvent( *pipe, MsgStream ))
    MsgStream.Reset();
50     pipe->fClosePipe();
    return 0;
}
```

SUBSTITUTE SHEET

```
FLAG fProcessClientEvent( MessagePipe &Pipe, TStream
&MsgStream )
{
    if (!Pipe.fGetMessage( MsgStream )) {
        cout << "Error reading message from pipe
5      DosErrorCode = " << Pipe.rcDosErrorCode() << endl;
        return FALSE;
    }

    10      CTMessageHeader Header;
        MsgStream >> Header;
        switch (Header.eType()) {
            case QUERY_CTID_STATUS:
                {
                    15      QueryCTIDStatusMsg StatusMsg( Header );
                        MsgStream >> *(QueryCTIDStatus*)&StatusMsg;
                        if (!fProcessQueryCTIDStatus( Pipe, StatusMsg ))
                            cout << "Error in fProcessQueryCTIDStatus,
20      SQLCODE = " << (long)ulGetSQLCode() << endl;
                        }
                        break;
                        case STORE_MONITOREVENT:
                            {
                                25      StoreMonitorEventMsg EventMsg( Header );
                                    MsgStream >> *(StoreMonitorEvent*)&EventMsg;
                                    if (!fProcessStoreMonitorEvent( Pipe, EventMsg
)) {
                                        cout << "Error in fProcessStoreMonitorEvent,
30      SQLCODE = " << (long)ulGetSQLCode() << endl;
                                    }
                                }
                                break;
                                case CLI_QUIT:
                                    return FALSE;
                                35      default:
                                    cout << "Unknown Command Received!" << endl;
                                    return FALSE;
                                }
                                return TRUE;
                                40      }
        }

    FLAG fProcessQueryCTIDStatus( MessagePipe &Pipe,
    45      QueryCTIDStatusMsg &CTID )
    {
        CTlicense Rec;
        CTIDStatusResultMsg ResultMsg;

        if (!fxlatCliCTID( CTID.CTID, CTID.CTID )) {
            50      cout << "Error converting client CTID to server
CTID" << endl;
                // Process error here.
            }
        }
```

SUBSTITUTE SHEET

```
ResultMsg.QueryResult = _fQueryLicense( &Rec,
CTID.CTID );

5      if (!ResultMsg.QueryResult) {
        ResultMsg.CTID = CTID.CTID;
        ResultMsg.Status =
CTLicStatus::ACTIVE;
        ResultMsg.PeriodDays = 2;
        ResultMsg.PeriodMinutes = 0;
10      ResultMsg.StolenFlag = FALSE;
        ResultMsg.SpecialProcess = 0;
        ResultMsg.Orgnum_n .fSetNull();
        ResultMsg.LastCallTS_n .fSetNull();
        ResultMsg.NextCallTS_n .fSetNull();
15      ResultMsg.NextCallClientTS_n .fSetNull();
        ResultMsg.ProductType .fSetNull();
      }
      else {
20      ResultMsg.CTID = Rec.CTID;
        ResultMsg.Status = Rec.LicStatus;
        ResultMsg.PeriodDays = Rec.PeriodDays;
        ResultMsg.PeriodMinutes = Rec.PeriodMinutes;
        ResultMsg.StolenFlag = Rec.StolenFlag ==
'y';
25      ResultMsg.SpecialProcess = Rec.SpecialProcess;
        ResultMsg.LastCallTS_n .Assign(
Rec.LastCallTS_N, DB_ISNULL( Rec.IsNull_LastCallTS ) );
        ResultMsg.NextCallTS_n .Assign(
30      Rec.NextCallTS_N, DB_ISNULL( Rec.IsNull_NextCallTS ) );
        ResultMsg.NextCallClientTS_n .Assign(
Rec.NextCallClientTS_N, DB_ISNULL(
Rec.IsNull_NextCallClientTS ) );
        if (DB_ISNULL( Rec.IsNull_Orgnum ))
35      ResultMsg.Orgnum_n .fSetNull();
        else
        ResultMsg.Orgnum_n = Rec.Orgnum_N;
        ResultMsg.ProductType = Rec.ProductType;
      }

40      cout << "SQLCODE = " << (long)ulGetSQLCode() << endl;

// Return Query results.
TStream Stream;
Stream << ResultMsg;
45      return Pipe.fSendMessage( Stream );
    }

50      FLAG fProcessStoreMonitorEvent( MessagePipe &Pipe,
StoreMonitorEventMsg &Msg )
    {
        StoreResultMsg ResultMsg;

// Prepare reply message.
55      ResultMsg.Result = TRUE;
```

SUBSTITUTE SHEET

```
// Prepare the monitorevent data.
    _CTmonitorEvent Rec;

    if (!fXlatCliCTID( (ULONG&)Rec.CTID, Msg.CTID )) {
5      cout << "Error converting client CTID to server
        CTID" << endl;
        // Process error here.
    }

10    _fCopyTStoDBVars( Rec.ServerTS, NULL,
        Msg.ServerTS, "ServerTS" );
    _fCopyTStoDBVars( Rec.ClientTS, NULL,
        Msg.ClientTS, "ClientTS" );
    _fCopyTStoDBVars( Rec.TelcoTS_N, &Rec.IsNull_TelcoTS,
15    Msg.TelcoTS_n, "TelcoTS" );

    Rec.DurationSec_N = Msg.DurationSec_n;
    Rec.IsNull_DurationSec = DB_NOT_NULL;

20    if (!Msg.CallerID_n) {
        Rec.IsNull_CallerID = DB_NULL;
    }
    else {
        Rec.IsNull_CallerID = DB_NOT_NULL;
25        strncpy( Rec.CallerID_N, Msg.CallerID_n, sizeof(
            Rec.CallerID_N ) );
    }

    Rec.LineNum = Msg.LineNum;

30    if (!Msg.LogFlag) {
        cout << "INVALID DATA ERROR: LogFlag is NULL,
            defaulting to FALSE" << endl;
        Rec.LogFlag = 'N';
    }
    else {
35        Rec.LogFlag = ((STRING)Msg.LogFlag)[0];
    }

    strncpy( Rec.EnvironmentID, Msg.EnvironmentID, sizeof(
40    Rec.EnvironmentID ) );

    Rec.ErrorCnt = Msg.ErrorCnt;

45    // Update the License Record.
    if (!fUpdateLicenseStatus( Msg )) {
        if (ulGetSQLCode() != 100) {
            cout << "DB2_ERROR: Error updating License
50            Table, CliCTID = " << Msg.CTID
                << " SQLCODE = " << (long)ulGetSQLCode() <<
            endl;
        }
    }

55    // Perform the insert.
```

SUBSTITUTE SHEET

```

    if (!_fInsertIntoMonitorEvent( &Rec )) {
        ResultMsg.Result = FALSE;
    }
    else {
5       if (Msg.StoreAsStolen) {
            if (!_fInsertIntoMonitorEventStolen( &Rec )) {
                ResultMsg.Result = FALSE;
            }
        }
10      if (Msg.StoreAsExpire) {
            if (!_fInsertIntoMonitorEventExpired( &Rec )) {
                ResultMsg.Result = FALSE;
            }
        }
15    }

    cout << "SQLCODE = " << (long)ulGetSQLCode() << endl;

    TStream Stream;
20    Stream << ResultMsg;
    if (Pipe.fSendMessage( Stream ) && ResultMsg.Result ==
TRUE) {
        DB.Commit();
        return TRUE;
25    }
    else {
        DB.Rollback();
        return FALSE;
30    }
}

FLAG fUpdateLicenseStatus( StoreMonitorEventMsg &Msg )
{
35    CTupdateLicenseStatus Rec;
    short dummy1; // Used to quiet the
Null validation below.

    fXlatCliCTID( (ULONG&)Rec.CTID, Msg.CTID );
40    strncpy( Rec.Status, Msg.LicenseStatus, sizeof(
Rec.Status ) );

    fCopyTStoDBVars( Rec.LastCallTS_n, &dummy1,
Msg.ServerTS, "LastCallTS" );
45    fCopyTStoDBVars( Rec.NextCallTS_n, &dummy1,
Msg.NextCallTS_n, "NextCallTS" );
    fCopyTStoDBVars( Rec.NextCallClientTS_n, &dummy1,
Msg.NextCallClientTS_n, "NextCallClientTS" );

50    if (!Msg.NextCallTS_n) strcpy( Rec.NextCallTS_n,
"0001-01-01-00.00.00.000000" );
    if (!Msg.NextCallClientTS_n) strcpy(
Rec.NextCallClientTS_n, "0001-01-01-00.00.00.000000" );

55    return _fUpdateLicenseStatus( &Rec );
}
```

SUBSTITUTE SHEET


```

}

FLAG_fCopyTStoDBVars( char *tsstring, short *indicator,
5 CTTimestamp &ts, STRING varname )
{
    if (!ts) {
        if (indicator == NULL) {
            cout << "INVALID DATA ERROR: " << varname << "
10 is NULL, forcing validation" << endl;
            ts.ForceValidate();
        }
        else {
            *indicator = DB_NULL;
            tsstring[0] = '\x0';
15 return FALSE;
        }
    }
    else if (!ts.fValidate()) {
        cout << "INVALID DATA ERROR: " << varname << " is
20 invalid, forcing validation - " << ts << endl;
        ts.ForceValidate();
    }

    if (indicator != NULL) *indicator = DB_NOT_NULL;
    ts.ToSTRING( tsstring );
    return TRUE;
}

30

#define INCL_NOPMAPI                // no PM in this program
#define INCL_DOS
35 #define INCL_BSE
#define INCL_DOSSEMAPHORES
#define INCL_DOSNMPPIPES
#include <os2.h>

40 #include <ctype.h>
#include <stdlib.h>
#include <iostream.h>
#include <fstream.h>

45 #include <server.h>

#include <MessagePipe.HPP>
#include <TModem.HPP>

50 #include "CT_Trans.H"

/*GLOBAL
VARIABLES******/

55 HEV hQuitSem;
```

SUBSTITUTE SHEET

```
// Temp, move to thread.
CltMsgPipeFactory *factory;
MessagePipe *pipe;

5  /*****
   **/

FLAG fLoadLineThreads( TModem&, PCSZ, PCSZ );
void _Optlink CT_CommandThread( PVOID );
10 FLAG fParseCmd( TPort &Port, TConnectInfo *CnctInfo,
   STRING buffer );

TPort::ComSettings ComSetting = {
15   "COM1",          // port name
   0,               // not used
   38400,           // bps
   8,               // data bits
   TPort::NO,       // no parity
20   TPort::ONE       // one stop bit
};

int main( int argc, char *argv[] )
{
25   APIRET rc;

   cout << "CompuTrace Server V0.99q" << endl;

   // Check arguments.
   if (argc != 4) {
30     cout << "Usage: server <pipe_name> <port_name>
<init_string>" << endl << endl;
     return 0;
   }

35   // Create quit semaphore.
   if ((rc = DosCreateEventSem( NULL, &hQuitSem, 0, FALSE
   )) != 0)
     return 1;

40   factory = new CltMsgPipeFactory( argv[1], 512 );

   // Load port server threads.
   TPort Port;
   TModem Modem = Port;
45   if (!fLoadLineThreads( Modem, argv[2], argv[3] ))
     return 2;

   cout << "Successfully connected to local modem" <<
50   endl;

   // Wait for quit signal.
   DosWaitEventSem( hQuitSem, SEM_INDEFINITE_WAIT );

   return 0;
55 }
```

SUBSTITUTE SHEET

```

//
// fLoadLineThreads: Loads the threads to operate a
5 server line. This function
// should be called for each server
line.
//
FLAG fLoadLineThreads( TModem &Modem, PCSZ port_str, PCSZ
10 init_str )
{
// Start port log.
// Port.LogOn();

// Open port.
15 ComSetting.port_name = port_str;
if (!Modem.Port().fOpenPort( ComSetting )) {
    cout << "Error openning port" << endl;
    return FALSE;
}

// Start the port manage thread.
if (!Modem.Port().fStartManageThread()) {
20     cout << "Thread execution error" << endl;
    return FALSE;
}

// Initialize the modem.
STRING result = Modem.strSendCommand( init_str, -1 );
30 if (strcmp( result, "OK" ) != 0) {
    cout << "Error initiallizing modem" << endl;
    return FALSE;
}

// Connect pipe to dbserver.
35 if (!factory->fCreatePipe( pipe )) return FALSE;
if (!pipe->fOpenPipe()) return FALSE;

// Start the command thread.
if (!Modem.Port().fStartCommandThread(
40 CT_CommandThread, (PVOID)&Modem )) {
    cout << "Thread execution error" << endl;
    Modem.Port().KillManageThread();
    return FALSE;
}

45 return TRUE;
}

//
50 // CT_CommandThread: Processes incoming data from a
server line.
//
void _Optlink CT_CommandThread( PVOID ptr )
55 {

```

SUBSTITUTE SHEET

```

    TModem &Modem = *(TModem*)ptr;           // Alias
    (should be optimized out by the compiler).

// Thread local variables
5   STRING result;
    TConnectInfo cnct_info;

    while (TRUE) {
        result = Modem.strGetString( -1 );
10   // Parse buffer for cmd.
        if (!fParseCmd( Modem.Port(), &cnct_info, result ))
        {
            memset( (PVOID)&cnct_info, '\x0', sizeof
15   cnct_info );
        }
    }

#define CND_DATE_FIELD      "DATE ="
20 #define CND_TIME_FIELD   "TIME ="
#define CND_NUMBER_FIELD   "NMBR ="

#define CND_NONUM_FIELD     "REASON FOR NO NUMBER:"
#define CND_NAME_FIELD      "CALLER NAME:"
25 #define CND_NONAME_FIELD  "REASON FOR NO NAME:"

//
// fParseCmd: called when a '\n' has been received, this
// function will process the string.
30 // Returns TRUE if a transaction is occurring,
// FALSE if the buffers should be cleared.
//

FLAG fParseCmd( TPort &Port, TConnectInfo *cnct_info,
35 STRING buffer )
{
    const char *index;

// Parse command.
40 if (strstr( buffer, "RING" ) != NULL) {
    cout << "Command parsed as RING" << endl;
}
else if ((index = strstr( buffer, CND_DATE_FIELD )) !=
45 NULL) {
    index += sizeof CND_DATE_FIELD;
    while (!isdigit( *index )) index++;
    // Grab the month.
    if (!isdigit( *index ) || !isdigit( *(index+1) ))
return FALSE;
50 cnct_info->cnd.month = (*index++ - '0') * 10;
cnct_info->cnd.month += *index++ - '0';
    // Grab the day.
    if (!isdigit( *index ) || !isdigit( *(index+1) ))
return FALSE;
55 cnct_info->cnd.day = (*index++ - '0') * 10;
```

SUBSTITUTE SHEET

```
cnct_info->cnd.day += *index++ - '0';

cout << buffer << endl;
}
5   else if ((index = strstr( buffer, CND_TIME_FIELD )) !=
NULL) {
    index += sizeof CND_TIME_FIELD;
    while (!isdigit( *index )) index++;
    // Grab the hour.
10   if (!isdigit( *index ) || !isdigit( *(index+1) ))
return FALSE;
    cnct_info->cnd.hour = (*index++ - '0') * 10;
    cnct_info->cnd.hour += *index++ - '0';
    // Grab the minute.
15   if (!isdigit( *index ) || !isdigit( *(index+1) ))
return FALSE;
    cnct_info->cnd.minute = (*index++ - '0') * 10;
    cnct_info->cnd.minute += *index++ - '0';

20   cout << buffer << endl;
}
    else if ((index = strstr( buffer, CND_NUMBER_FIELD ))
!= NULL) {
    index += sizeof CND_NUMBER_FIELD;
    while (isspace( *index )) index++;
    // Grab the number.
    for (int i = 0; i < CND_NUM_MAXLEN; i++) {
        if (index[i] == '\x0' || index[i] == '\r') {
30         cnct_info->cnd.number[i] = '\x0';
        break;
        }
        else {
            cnct_info->cnd.number[i] = index[i];
        }
    }
35   cout << buffer << endl;
}
    else if (strstr( buffer, CND_NONUM_FIELD ) != NULL) {
    index += sizeof CND_NONUM_FIELD;
40   // Grab the string.
    while (isspace( *index )) index++;
    for (int i = 0; i < CND_NUM_MAXLEN; i++) {
        if (index[i] == '\x0' || index[i] == '\r') {
45         cnct_info->cnd.number[i] = '\x0';
        break;
        }
        else {
            cnct_info->cnd.number[i] = index[i];
        }
    }
50   }

    cout << buffer << endl;
}
    else if (strstr( buffer, CND_NAME_FIELD ) != NULL) {
55   index += sizeof CND_NAME_FIELD;
```

SUBSTITUTE SHEET

```

// Grab the name.
while (isspace( *index )) index++;
for (int i = 0; i < CND_NAME_MAXLEN; i++) {
5   if (index[i] == '\x0' || Index[i] == '\r') {
        cnct_info->cnd.name[i] = '\x0';
        break;
    }
    else {
10        cnct_info->cnd.name[i] = index[i];
    }
}

cout << buffer << endl;
15 }
else if (strstr( buffer, CND_NONAME_FIELD ) != NULL)
{
    index += sizeof CND_NONAME_FIELD;
    // Grab the string.
    while (isspace( *index )) index++;
20   for (int i = 0; i < CND_NAME_MAXLEN; i++) {
        if (index[i] == '\x0' || Index[i] == '\r') {
            cnct_info->cnd.name[i] = '\x0';
            break;
        }
        else {
25            cnct_info->cnd.name[i] = index[i];
        }
    }
}

30   cout << buffer << endl;
}
else if (strstr( buffer, "CONNECT" ) != NULL) {
    cout << "Command parsed as CONNECT" << endl;
35   SntlConnect( Port, *pipe, cnct_info );
    return FALSE;
}
else if (strstr( buffer, "NO CARRIER" ) != NULL) {
40   cout << "Command parsed as NO CARRIER" << endl;
    return FALSE;
}
else if (strstr( buffer, "OK" ) != NULL) {
    cout << "Command parsed as OK" << endl;
45   return FALSE;
}
else if (strstr( buffer, "ERROR" ) != NULL) {
    cout << "Command parsed as ERROR" << endl;
    return FALSE;
}
50   else {
        cout << "Unknown command received: " << buffer <<
endl;
        return FALSE;
    }
55   return TRUE;
}
```

SUBSTITUTE SHEET

```
        return TRUE;
    }
}

5  FLAG CTOrgnum::fSetIndex( UINT num )
    {
        if (num > 9999) {
            return FALSE;
        }
10     else {
        value[ORGNUM_PREFIX_SIZE + 0] = (num%10000) / 1000
+ '0';
        value[ORGNUM_PREFIX_SIZE + 1] = (num%1000) / 100 +
'0';
15     value[ORGNUM_PREFIX_SIZE + 2] = (num%100) / 10 +
'0';
        value[ORGNUM_PREFIX_SIZE + 3] = (num % 10) + '0';
    }
}

20  FLAG CTOrgnum::fGetPrefix( char *str ) const
    {
        if (strlen( str ) != ORGNUM_PREFIX_SIZE) {
25     return FALSE;
        }
        else {
            str[0] = value[0];
            str[1] = value[1];
            str[2] = value[2];
30     str[3] = value[3];
            str[4] = '\x0';
        }
    }

35  FLAG CTOrgnum::fGetIndex( UINT &i ) const
    {
        i = atoi( &(value[ORGNUM_PREFIX_SIZE]) );
        return TRUE;
    }

40  FLAG CTOrgnum::fGeneratePrefix( STRING org_name )
    {
        char pre[ORGNUM_PREFIX_SIZE];

45     // Grab first four alphanum characters.
        for (int i = 0, j = 0; i < ORGNUM_PREFIX_SIZE;) {
            if (isalnum( orgname[j++] )) pre[i];
        }
50     *****/

        //*****
        //*****
        //
55     // iostream stream operators.
```

SUBSTITUTE SHEET

```
USHORT num;

buf >> *(TNull*)&lic;
if (!lic) return buf;
5   else {
      buf >> num;
      lic.value = CTLicStatus::VALUE( num );
      return buf;
10  }

TStream& operator << ( TStream &buf, const CTOrgnum &num
)
{
15   buf << *(TNull*)&num;
      if (!num) return buf;
      else return buf.Put( PVOID( num.value ), sizeof(
num.value ) );
20 }

TStream& operator >> ( TStream &buf, CTOrgnum &num )
{
25   buf >> *(TNull*)&num;
      if (!num) return buf;
      else return buf.Get( num.value, sizeof( num.value ) );
}

TStream& operator << ( TStream &buf, const CTMonitorEvent
&event )
30 {
    return buf << event.CTID
        << event.ServerTS
        << event.ClientTS
35     << event.TelcoTS_n
        << event.DurationSec_n
        << event.CallerID_n
        << event.LineNum
        << event.LogFlag
40     << event.EnvironmentID
        << event.ErrorCnt;
}

TStream& operator >> ( TStream &buf, CTMonitorEvent
&event )
45 {
    return buf >> event.CTID
        >> event.ServerTS
        >> event.ClientTS
50     >> event.TelcoTS_n
        >> event.DurationSec_n
        >> event.CallerID_n
        >> event.LineNum
        >> event.LogFlag
55     >> event.EnvironmentID
}
```

SUBSTITUTE SHEET


```
{
    ULONG post_count;

    DosRequestMutexSem( hBufSem, SEM_INDEFINITE_WAIT );
5    head = 0;
    tail = CT_BUFFER_MAXLEN;
    DosResetEventSem( hReleaseGetSem, &post_count );
    DosReleaseMutexSem( hBufSem );
}

10 FLAG CT_Buffer::fPutChar( char ch )
{
    FLAG ret_val;

15    // Get ownership of the semaphore.
    rc = DosRequestMutexSem( hBufSem, SEM_INDEFINITE_WAIT );
    if (rc) return FALSE;

20    // First check that the log buffer hasn't overflowed.
    if (!fIsFull()) {
        // Store the char, update head, signal the event.
        buffer[head] = ch;
        head = IncBufPtr( head );
25        DosPostEventSem( hReleaseGetSem );
        ret_val = TRUE;
    }
    else ret_val = FALSE;

30    // Release the semaphore.
    DosReleaseMutexSem( hBufSem );

    return ret_val;
}

35 FLAG CT_Buffer::fGetChar( char &ch )
{
    ULONG post_count;
    FLAG ret_val;

40    // If empty wait for timeout.
    if (fIsEmpty()) DosWaitEventSem( hReleaseGetSem,
SEM_INDEFINITE_WAIT );

45    // Get ownership of the semaphore.
    rc = DosRequestMutexSem( hBufSem, SEM_INDEFINITE_WAIT );
    if (rc) return FALSE;

50    if (!fIsEmpty()) {
        // Fetch the char, update tail.
        tail = IncBufPtr( tail );
        ch = buffer[tail];
        ret_val = TRUE;
55    }
```

SUBSTITUTE SHEET

```

    return TRUE;
}

#define INCL_DOSNMPPIPES
#include <os2.h>

#include <MessagePipe.HPP>

//*****
// SvrMsgPipeFactory Implementation.
//*****

SvrMsgPipeFactory::SvrMsgPipeFactory( PCSZ name, UINT
msg_len, UINT pipe_len )
: MsgPipeFactory( msg_len ),
  pipe_name( name ),
  pipe_len( pipe_len )
{}

FLAG SvrMsgPipeFactory::fCreatePipe( MessagePipe *ppipe
)
{
    ppipe = new MessagePipe( this );
    return TRUE;
}

FLAG SvrMsgPipeFactory::fDestroyPipe( MessagePipe *ppipe
)
{
    delete ppipe;
    return TRUE;
}

FLAG SvrMsgPipeFactory::fOpenPipe( MessagePipe *pipe )
{
    HPIPE hPipe;

    // Create and connect the named pipe.
    pipe->rc = DosCreateNPipe( (PSZ)pipe_name, &hPipe,
    NP_NOWRITEBEHIND ;
45 Data sent to remote pipes immediatly. //
    NP_ACCESS_DUPLEX, //
    Two-way client/server communications. //
    NP_WAIT ; //
50 I/O to pipe blocked until data available. //
    NP_TYPE_MESSAGE ; //
    NP_READMODE_MESSAGE ; //
    0x00FF, //
55 Infinite number of allowed instances of this pipe. //

```

SUBSTITUTE SHEET

```
        return TRUE;
    }

5  FLAG CltMsgPipeFactory::fOpenPipe( MessagePipe *pipe )
    {
        HPIPE hPipe;
        ULONG ulAction;

10     pipe->rc = DosOpen( pipe_name, &hPipe, &ulAction, 0,
                        FILE_NORMAL, FILE_OPEN,
                        OPEN_ACCESS_READWRITE |
                        OPEN_SHARE_DENYNONE,
                        (PEAOP2) NULL );
15     if (pipe->rc) return FALSE;

        pipe->SetHandle( hPipe );
        return TRUE;
    }

20  FLAG CltMsgPipeFactory::fClosePipe( MessagePipe *pipe )
    {
        HPIPE hPipe = pipe->GetHandle();

25     // Wait till the pipe is empty.
        pipe->rc = DosResetBuffer( hPipe );
        if (pipe->rc) return FALSE;
        // Close the pipe handle.
        rc = DosClose( hPipe );
30     if (pipe->rc) return FALSE;

        return TRUE;
    }

35  /*******
    *****
    // MessagePipe Implementation
    /*******
    *****

40  MessagePipe::MessagePipe( MsgPipeFactory *mom )
        : factory( mom )
    {
        factory->InitPipe( this );
    }

45  MessagePipe::~MessagePipe()
    {
        factory->DeinitPipe( this );
    }

50  FLAG MessagePipe::fOpenPipe()
    {
        return factory->fOpenPipe( this );
    }

55
```

SUBSTITUTE SHEET

```
#include <os2.h>
#endif

#include <ctype.h>
#include <Objects.HPP>

//*****
//
// TFlag members.
//

TFlag::TFlag()
: TNull( TRUE )
{}

TFlag::TFlag( FLAG flag )
: value( (flag != FALSE) ),
  TNull( FALSE )
{}

TFlag::~TFlag()
{
#ifdef DEBUG
    fSetNull();
    value = UNINIT_DATA;
#endif
}

//*****
//
// TTimestamp members.
//

const UINT TTimestamp::TSStringLen = 27;

TTimestamp::TTimestamp()
: TNull( TRUE )
{
#ifdef DEBUG
    Year = Month = Day = Hour = Minute = Second =
    Millisec = UNINIT_DATA;
#endif
}

TTimestamp::TTimestamp( USHORT yr, UCHAR mo, UCHAR dy,
                        UCHAR hr, UCHAR mn, UCHAR sc,
                        USHORT ms )
: Year( yr ),
  Month( mo ),
  Day( dy ),
  Hour( hr ),
  Minute( mn ),
```

SUBSTITUTE SHEET

```
        Second( sc ),
        Millisec( ms ),
        TNull( FALSE )
    {}
5
Timestamp::~Timestamp()
{
    #ifdef DEBUG
        fSetNull();
10        Year = Month = Day = Hour = Minute = Second =
        Millisec = UNINIT_DATA;
        #endif
    }

15    FLAG Timestamp::fValidate() const
    {
        if (fIsNull()) return FALSE;

        // Check year.
        if (!Year || Year > 9999) return FALSE;
        // Check month and day.
        if (!Day) return FALSE;
        switch (Month) {
25            case 1:
                if (Day > 31) return FALSE;
                break;
            case 2:
                if (Year % 4 == 0 && Year % 100 != 0) //
30                Check for a leapyear.
                if (Day > 29) return FALSE;
                else
                if (Day > 28) return FALSE;
                break;
            case 3:
35                if (Day > 31) return FALSE;
                break;
            case 4:
                if (Day > 30) return FALSE;
                break;
40            case 5:
                if (Day > 31) return FALSE;
                break;
            case 6:
                if (Day > 30) return FALSE;
45                break;
            case 7:
                if (Day > 31) return FALSE;
                break;
            case 8:
50                if (Day > 31) return FALSE;
                break;
            case 9:
                if (Day > 30) return FALSE;
                break;
55            case 10:
```

SUBSTITUTE SHEET

```

        if (Day > 31) return FALSE;
        break;
    case 11:
        if (Day > 30) return FALSE;
        break;
    case 12:
        if (Day > 31) return FALSE;
        break;
    default:
        return FALSE;
    }
    // Check hours.
    if (Hour > 23) {
        if (Hour > 24 || Minute || Second || Millisec)
            return FALSE;
    }
    // Check minutes, seconds and milliseconds.
    if (Minute > 59 || Second > 59 || Millisec > 999)
        return FALSE;

    return TRUE;
}

void TTimestamp::ForceValidate()
{
    setNotNull();
    Year = Month = Day = 1;
    Hour = Minute = Second = Millisec = 0;
}

FLAG TTimestamp::IsValidTSString( STRING ts )
{
    if (
        isdigit( ts[0] )           // Check Year.
        && isdigit( ts[1] )
        && isdigit( ts[2] )
        && isdigit( ts[3] )
        && ts[4] == '-'
        && isdigit( ts[5] )       // Check Month.
        && isdigit( ts[6] )
        && ts[7] == '-'
        && isdigit( ts[8] )       // Check Day.
        && isdigit( ts[9] )
        && ts[10] == '-'
        && isdigit( ts[11] )      // Check Hour.
        && isdigit( ts[12] )
        && ts[13] == '.'
        && isdigit( ts[14] )      // Check Minute.
        && isdigit( ts[15] )
        && ts[16] == '.'
        && isdigit( ts[17] )      // Check Second.
        && isdigit( ts[18] )
        && ts[19] == '.'
        && isdigit( ts[20] )      // Check Millisec.
        && isdigit( ts[21] )
        && isdigit( ts[22] )
    )

```

SUBSTITUTE SHEET

```

    && isdigit( ts[23] )
    && isdigit( ts[24] )
    && isdigit( ts[25] )
    && ts[26] == '\x0'
5      return TRUE;
      else return FALSE;
  }

10  Timestamp& Timestamp::Assign( const Timestamp &ts )
  {
    if (!ts) {
      fSetNull();
    }
    else {
15      setNotNull();
      Year = ts.Year;
      Month = ts.Month;
      Day = ts.Day;
      Hour = ts.Hour;
20      Minute = ts.Minute;
      Second = ts.Second;
      Millisec = ts.Millisec;
    }
    return (*this);
25  }

Timestamp& Timestamp::Assign( USHORT yr, UCHAR mo,
UCHAR dy,
                                UCHAR hr, UCHAR mn, UCHAR
30  sc, USHORT ms )
  {
    setNotNull();

    Year = yr;
35    Month = mo;
    Day = dy;
    Hour = hr;
    Minute = mn;
    Second = sc;
40    Millisec = ms;

    return (*this);
  }

45  Timestamp& Timestamp::Assign( STRING ts, FLAG isnull )
  {
    unsigned num;

    if (isnull) {
50      fSetNull();
      return *this;
    }

    setNotNull();
55  }
```

SUBSTITUTE SHEET

```
    ASSERT( fIsValidTSString( ts ) );

    /* Convert year */
    num = (ts[0] - '0') * 1000;
5    num += (ts[1] - '0') * 100;
    num += (ts[2] - '0') * 10;
    num += (ts[3] - '0');
    Year = USHORT( num );
10    /* Convert month */
    num = (ts[5] - '0') * 10;
    num += (ts[6] - '0');
    Month = UCHAR( num );
    /* Convert day */
15    num = (ts[8] - '0') * 10;
    num += (ts[9] - '0');
    Day = UCHAR( num );
    /* Convert hour */
    num = (ts[11] - '0') * 10;
20    num += (ts[12] - '0');
    Hour = UCHAR( num );
    /* Convert minute */
    num = (ts[14] - '0') * 10;
    num += (ts[15] - '0');
25    Minute = UCHAR( num );
    /* Convert second */
    num = (ts[17] - '0') * 10;
    num += (ts[18] - '0');
    Second = UCHAR( num );
30    /* Convert millisec */
    num = (ts[20] - '0') * 100;
    num += (ts[21] - '0') * 10;
    num += (ts[22] - '0');
    Millisec = USHORT( num );
35    return *this;
}

#ifdef __OS2__
40    TTimestamp& TTimestamp::Assign( const DATETIME &Date )
    {
        setNotNull();

        Year = Date.year;
        Month = Date.month;
45        Day = Date.y;
        Hour = Date.hours;
        Minute = Date.minutes;
        Second = Date.seconds;
        Millisec = Date.hundredths * 10;
50        return (*this);
    }
#endif // __OS2__
55    STRING TTimestamp::ToSTRING( char *ts ) const
```

SUBSTITUTE SHEET


```
{
    unsigned num;

    /* Convert year */
5    num = Year;
    ts[0] = (num%10000) / 1000 + '0';
    ts[1] = (num%1000) / 100 + '0';
    ts[2] = (num%100) / 10 + '0';
    ts[3] = (num % 10) + '0';
10   ts[4] = '-';
    /* Convert month */
    num = Month;
    ts[5] = (num%100) / 10 + '0';
    ts[6] = (num % 10) + '0';
15   ts[7] = '-';
    /* Convert day */
    num = Day;
    ts[8] = (num%100) / 10 + '0';
    ts[9] = (num % 10) + '0';
20   ts[10] = '-';
    /* Convert hour */
    num = Hour;
    ts[11] = (num%100) / 10 + '0';
    ts[12] = (num % 10) + '0';
25   ts[13] = '.';
    /* Convert minute */
    num = Minute;
    ts[14] = (num%100) / 10 + '0';
    ts[15] = (num % 10) + '0';
30   ts[16] = '.';
    /* Convert second */
    num = Second;
    ts[17] = (num%100) / 10 + '0';
    ts[18] = (num % 10) + '0';
35   ts[19] = '.';
    /* Convert millisec */
    num = Millisec;
    ts[20] = (num%1000) / 100 + '0';
    ts[21] = (num%100) / 10 + '0';
40   ts[22] = (num % 10) + '0';
    ts[23] = '0';
    ts[24] = '0';
    ts[25] = '0';

45   ts[26] = '\\x0';

    return ts;
}

50  FLAG TTimestamp::operator > ( const TTimestamp &ts )
    const
    {
        useAsValue();

55   if (Year > ts.Year) return TRUE;
```

SUBSTITUTE SHEET

```

5      else if (Year == ts.Year) {
        if (Month > ts.Month) return TRUE;
        else if (Month == ts.Month) {
          if (Day > ts.Day) return TRUE;
          else if (Day == ts.Day) {
            if (Hour > ts.Hour) return TRUE;
            else if (Hour == ts.Hour) {
              if (Minute > ts.Minute) return TRUE;
              else if (Minute == ts.Minute) {
                if (Second > ts.Second) return TRUE;
                else if (Second == ts.Second) {
                  if (Millisec > ts.Millisec) return
10      TRUE;
                  else return FALSE;
15      }
                }
              }
            }
          }
        }
20    }
    return FALSE;
}

25  FLAG TTimestamp::operator >= ( const TTimestamp &ts )
    const
    {
      return (*this > ts || *this == ts);
    }

30  FLAG TTimestamp::operator == ( const TTimestamp &ts )
    const
    {
      useAsValue();
35      if (Year == ts.Year &&
          Month == ts.Month &&
          Day == ts.Day &&
          Hour == ts.Hour &&
          Minute == ts.Minute &&
          Second == ts.Second &&
          Millisec == ts.Millisec) {
40      return TRUE;
      }
      else {
45      return FALSE;
      }
    }

50  // Date and time add function.
    TTimestamp& TTimestamp::AddToDate( UINT yr, UINT mon,
    UINT day,
                                UINT hr, UINT min,
    UINT sec, UINT ms )
    {
55      if (!fIsNull()) {
```

SUBSTITUTE SHEET

```

    ms += Millisec;
    sec += Second;
    min += Minute;
5    hr += Hour;
    day += Day;
    mon += Month;
    yr += Year;
}

10    // Adjust and carry ms.
    while (ms > usMaxMillisec()) {
        ms -= usMaxMillisec() + 1;
        sec++;
    }

15    // Adjust and carry sec.
    while (sec > usMaxSecond()) {
        sec -= usMaxSecond() + 1;
        min++;
    }

20    // Adjust and carry min.
    while (min > usMaxMinute()) {
        min -= usMaxMinute() + 1;
        hr++;
    }

25    // Adjust and carry hr.
    while (hr > usMaxHour()) {
        hr -= usMaxHour() + 1;
        day++;
    }

30    // Adjust and carry mon (day adjust is dependent on mon
    and yr).
    while (mon > usMaxMonth()) {
        mon -= usMaxMonth();
        yr++;
    }

35    // Now adjust and carry day now that yr and mon is known.
    while (day > usMaxDay( yr, mon )) {
        day -= usMaxDay( yr, mon );
        mon++;
    }

40    if (mon > usMaxMonth()) {
        mon -= usMaxMonth();
        yr++;
    }

    }

45    // Copy new values to members.

    Assign( yr, mon, day, hr, min, sec, ms );

50    CHECK( fValidate() );
    return *this;
}

// static member.
55    USHORT TTimestamp::usMaxDay( USHORT year, USHORT month )
```

SUBSTITUTE SHEET

```
{
    switch (month) {
        case 1:                // Jan.
            return 31;
5
        case 2:                // Feb.
            return fIsLeapYear( year ) ? 29 : 28;
        case 3:                // Mar.
10
            return 31;
        case 4:                // Apr.
            return 30;
        case 5:                // May.
15
            return 31;
        case 6:                // Jun.
20
            return 30;
        case 7:                // Jul.
            return 31;
        case 8:                // Aug.
25
            return 31;
        case 9:                // Sep.
            return 30;
        case 10:               // Oct.
30
            return 31;
        case 11:               // Nov.
            return 30;
35
        case 12:               // Dec.
            return 31;

        // default:
40
        // BOILERPLATE;
    }
}

//*****
45
//
// TStream stream operators.
//
TStream& operator << ( TStream &buf, const TFlag &flag )
50
{
    if (!flag) return buf << FLAG( TRUE );
    else return buf << FLAG( FALSE ) << flag.value;
}

55
TStream& operator >> ( TStream &buf, TFlag &flag )
```

SUBSTITUTE SHEET

```

    {
        buf >> *(TNull*)&flag;
        if (flag.isNull() == FALSE)
            buf >> flag.value;
5         return buf;
    }

TStream& operator << ( TStream &buf, const TTimestamp &ts
10 )
{
    if (!ts) return buf << FLAG( TRUE );
    else {
        return buf << FLAG( FALSE )
15             << ts.Year
             << ts.Month
             << ts.Day
             << ts.Hour
             << ts.Minute
20             << ts.Second
             << ts.Millisecond;
    }
}

TStream& operator >> ( TStream &buf, TTimestamp &ts )
25 {
    buf >> *(TNull*)&ts;
    if (!ts) {
        return buf;
    }
    else {
30         return buf >> ts.Year
                    >> ts.Month
                    >> ts.Day
                    >> ts.Hour
                    >> ts.Minute
35         >> ts.Second
                    >> ts.Millisecond;
    }
}

40 //*****
//
// iostream friend function members.
45 //

ostream& operator << ( ostream &os, const TFlag &flag )
{
    if (!flag) return os << NULL_TOK;
50     else return os << (STRING)flag;
}

/*****
55 istream& operator << ( istream &is, TFlag &flag )
{
```

SUBSTITUTE SHEET

```
char ch, buffer[12];

is >> ws;                                     // Extract leading
whitespace.

5      for (int i = 0; i < sizeof( buffer ); i++) {
          is >> buffer[i];
          if (!isalpha( buffer[i] )) break;
      }
10     if (i == sizeof( buffer ) ASSERT( FALSE );
        buffer[i] = '\x0';

        if (strcmp( buffer, NULL_TOK) == 0) {
15         fSetNull();
        }
        else if (strcmp( buffer, TRUE_TOK) == 0) {
            Assign( TRUE );
        }
20     else if (strcmp( buffer, FALSE_TOK) == 0) {
            Assign( FALSE );
        }
        else ASSERT( FALSE );

25     return is;
    }
    *****/

30     ostream& operator << ( ostream &os, const TTimestamp &ts
    )
    {
        char tsstring[TTimestamp::TSStringLen];
        if (!ts) return os << "NULL";
        else return os << ts.ToSTRING( tsstring );
35    }

#define INCL_NOPMAPI                          // no PM in this program
#define INCL_DOS
40    // #define INCL_BSE
    // #define INCL_DOSSEMAPHORES
    #include <os2.h>

    #include <usertype.h>
45    #include <TModem.HPP>

TModem::TModem( TPort &_port )
    : port( _port )
    {}

50    TModem::RC TModem::rcSendCommand( STRING, ULONG timeout )
    {
        NOTIMPLEMENTED;
    }

55
```

SUBSTITUTE SHEET

```
STRING TModem::strSendCommand( STRING str, ULONG timeout
)
{
    port.fWritePort( str );
    port.fPutChar( '\r' );
5    STRING result = strGetString( timeout );
    if (strcmp( str, result ) == 0) {
        return strGetString( timeout );
    }
10    else {
        return result;
    }
}

15 STRING TModem::strGetString( ULONG timeout )
{
    UINT i = 0;
    last_result[0] = '\x0';

20    // Eat Leading CR/NL.
    while (!port.fGetChar( last_result[i] )
           || last_result[i] == '\r'
           || last_result[i] == '\n') {}
    i++;
    // (already got 1 char ok)
25    // Grab text until a CR/NL.
    while (port.fGetChar( last_result[i] )
           && last_result[i] != '\n'
           && last_result[i] != '\r'
           && i <= sizeof( last_result )) {
30        i++;
    }
    last_result[i] = '\x0';           // Null terminate
    buffer.
    return last_result;
35 }

#include <TObject.HPP>

//*****
40 *****
//
// TObject members.
//

45 TObject::~TObject()
{}

//*****
50 *****
//
// TNull members.
//

TNull::TNull( FLAG is_null )
55 : isnull( is_null )
```

SUBSTITUTE SHEET

```
{  
FLAG TNull::fSetNull()  
{  
5      isnull = TRUE;  
      return TRUE;  
}  
  
10      #define INCL_NOPMAPI                // no PM in this program  
      #define INCL_DOS  
      #define INCL_BSE  
      #define INCL_DOSSEMAPHORES  
15      #define INCL_DOSNMPIPES  
      #include <os2.h>  
  
      #include <usertype.h>  
      #include "TPacket.HPP"  
20  
      TPacket::TPacket( TPort& p )  
          : Port( p ),  
            text_length( 0 ),  
            state( TRANS_NULL )  
25      {}  
  
      TPacket::TRANS_STATE TPacket::rGetPacket()  
      {  
30          enq_count = 0;  
          nak_count = 0;  
          text_length = 0;  
  
          if (state != TRANS_NULL) return TRANS_NULL;  
35      // Enquiry Loop.  
          while (fSendENQ())  
              {  
                  if ((state = rReceivePacket()) == TRANS_NAK)  
40                      {  
                          while (fSendNAK())  
                              if ((state = rReceivePacket()) == TRANS_ACK)  
                                  {  
45                                      fSendACK();  
                                      return state;  
                                  }  
                      }  
  
                  else if (state == TRANS_ACK)  
50                      {  
                          fSendACK();  
                          return state;  
                      }  
              }  
55      fSendEOT();
```

SUBSTITUTE SHEET


```
        return state;
    }

5    TPacket::TRANS_STATE TPacket::rReceivePacket()
    {
        char ch;
        int i=0,j;

10       // Get STX.
        if (!Port.fGetChar( ch ))
            return TRANS_ETO;
        // packet_text[i++] = ch;
        if (ch != STX)
15         return TRANS_NAK;

        // Get Length.
        if (!Port.fGetChar( ch ))
            return TRANS_NAK;
20       // packet_text[i++] = ch;

        text_length = (USHORT)ch;

        if (!Port.fGetChar( ch ))
25         return TRANS_NAK;
        // packet_text[i++] = ch;

        text_length = (USHORT)(ch << 8) + text_length;

30       if (text_length > MAX_TEXT_LEN)
            return TRANS_NAK;

        // Get Text.

35       for (j=0 ; j < text_length; j++ )
        {
            if ( Port.fGetChar( ch ))
                packet_text[ j ] = ch;

40             else
                return ( TRANS_NAK );
        }

        // Get ETX.
45       if ( Port.fGetChar( ch ))
        {
            if ( ch == ETX )
                ;
            // packet_text[ i++ ] = ch;
50             else
                return ( TRANS_NAK );
        }
        else
55         {
```

SUBSTITUTE SHEET

```
        return ( TRANS_NAK );
    }

    // Get LRC.
5    if (!Port.fGetChar( ch ))
        return TRANS_NAK;
    // packet_text[i++] = ch;
    return TRANS_ACK;
}

10    UINT TPacket::cbCopyText( PVOID ptr, UINT len )
    {
        len = len < text_length ? len : text_length;
        memcpy( ptr, packet_text, len );
15        return len;
    }

    FLAG TPacket::fSendENQ()
    {
20        char enq = ENQ;

        enq_count++;
        if (enq_count > MAX_ENQ) return FALSE;

25        Port.FlushInputBuffer();
        return Port.fWritePort( &enq, 1 );
    }

    FLAG TPacket::fSendACK()
30    {
        char ack = ACK;
        Port.FlushInputBuffer();
        return Port.fWritePort( &ack, 1 );
    }

35    FLAG TPacket::fSendNAK()
    {
        char nak = NAK;

40        nak_count++;
        if (nak_count > MAX_NAK) return FALSE;

        Port.FlushInputBuffer();
        return Port.fWritePort( &nak, 1 );
45    }

    FLAG TPacket::fSendEOT()
    {
50        char eot = EOT;
        return Port.fWritePort( &eot, 1 );
    }

55    #define INCL_NOPMAPI                // no PM in this program
    #define INCL_DOS
```

SUBSTITUTE SHEET

```
#define INCL_BSE
#define INCL_DOSSEMAPHORES
#define INCL_DOSNMPPIPES
#define INCL_DOSDEVIOTCL
5 #include <os2.h>

#define _THREADS // This implemetation is
multi-threaded.

10 #include <process.h>
#include <string.h>
#include <stdlib.h>

#include "TPort.HPP"

15 TPort::TPort()
: manage_thread( -1 ),
log_flag( FALSE )
{}

20 TPort::~TPort()
{
while (manage_thread != -1) {
25 KillManageThread();
DosSleep( 1000 ); // Wait 1 second.
}
}

30 FLAG TPort::fOpenPort( const ComSettings &settings )
{
LINECONTROL lctl;
DCBINFO dcb;
ULONG ulAction;
35 ULONG ulPio, ulDio;
ULONG cbTrans;

// Open the port.
rc = DosOpen( settings.port_name, &hPort, &ulAction,
0, 0, OPEN_ACTION_OPEN_IF_EXISTS,
40 OPEN_FLAGS_WRITE_THROUGH |
OPEN_ACCESS_READWRITE | OPEN_SHARE_DENYREADWRITE, NULL );
if (rc) return FALSE;

// Set the line speed.
45 ulPio = sizeof( settings.bps );
rc = DosDevIOctl( hPort, IOCTL_ASYNC,
ASYNC_SETBAUDRATE, (PVOID)&settings.bps,
ulPio, &ulPio, NULL, 0, NULL );
if (rc) {
50 DosClose( hPort );
return FALSE;
}

// Set the line characteristics.
55 lctl.bDataBits = settings.data_bits;
```

SUBSTITUTE SHEET

```
        lctl.bParity = (BYTE)settings.parity;
        lctl.bStopBits = (BYTE)settings.stop_bits;
        ulPio = sizeof lctl;
        rc = DosDevIOctl( hPort, IOCTL_ASYNC,
5      ASYNC_SETLINECTRL, &lctl, ulPio, &ulPio, NULL, 0, NULL );
        if (rc) {
            DosClose( hPort );
            return FALSE;
        }
10
        // Set the flow control.
        ulDio = sizeof dcb;
        rc = DosDevIOctl( hPort, IOCTL_ASYNC,
        ASYNC_GETDCBINFO, NULL, 0, NULL, &dcb, ulDio, &ulDio );
15      if (rc) {
            DosClose( hPort );
            return FALSE;
        }
        /*****
20      *****/
        dcb.usReadTimeout = 100;

        dcb.fbCtlHndShake = MODE_CTS_HANDSHAKE;    // flags1 =
25      00001000

        dcb.fbFlowReplace &= 0x30;                // flags2 =
        00??0000
        dcb.fbFlowReplace |= MODE_RTS_HANDSHAKE;   // flags2 =
30      10??0000

        dcb.fbTimeout &= 0xF8;                    // flags3 =
        ?????000
        dcb.fbTimeout |= MODE_WAIT_READ_TIMEOUT;  // flags3 =
35      ?????100
        *****/
        *****/
        dcb.usReadTimeout = 300;
        dcb.fbCtlHndShake = MODE_CTS_HANDSHAKE;
        dcb.fbFlowReplace = MODE_RTS_HANDSHAKE;
40      dcb.fbTimeout = MODE_NO_WRITE_TIMEOUT |
        MODE_WAIT_READ_TIMEOUT;

        rc = DosDevIOctl( hPort, IOCTL_ASYNC,
45      ASYNC_SETDCBINFO, &dcb, ulPio, &ulPio, NULL, 0, NULL );
        if (rc) {
            DosClose( hPort );
            return FALSE;
        }
50
        fRaiseDTR();

        return TRUE;
    }
55
    FLAG TPort::fClosePort()
```

SUBSTITUTE SHEET

```
{
    rc = DosClose( hPort );
    if (rc) return FALSE;
    else return TRUE;
}

void TPort::FlushInputBuffer()
{
    BYTE cmd;
    by API. // Scratch, Needed
    ULONG len;
    by API. // Scratch, Needed

    rc = DosDevIOctl( hPort, IOCTL_GENERAL,
DEV_FLUSHINPUT, &cmd, sizeof( cmd ), &len,
&cmd, sizeof( cmd ), &len );

    DosSleep(10); // Timing Kludge - Give the
Device Driver // time to flush buffer before
resetting // semaphore stuff.
    buffer.Flush();
}

void TPort::FlushOutputBuffer()
{
    BYTE cmd;
    by API. // Scratch, Needed
    ULONG len;
    by API. // Scratch, Needed

    rc = DosDevIOctl( hPort, IOCTL_GENERAL,
DEV_FLUSHOUTPUT, &cmd, sizeof( cmd ), &len,
&cmd, sizeof( cmd ), &len );
}

FLAG TPort::fReadPort( PVOID buf, UINT &len )
{
    for (int i = 0; i < len; i++) {
        if (buffer.fIsEmpty()) {
            len = i;
            return TRUE;
        }
        else buffer.fGetChar( ((char*)buf)[i] );
    }
    return TRUE;
}

FLAG TPort::fWritePort( PVOID buf, UINT len )
{
    ULONG cbWritten;

    rc = DosWrite( hPort, buf, len, &cbWritten );
    if (rc) return FALSE;
```

SUBSTITUTE SHEET

```
        else return TRUE;
    }

    FLAG TPort::fDropDTR()
5    {
        ULONG ulPio, ulDio;
        MODEMSTATUS ms;
        ULONG com_err;

10        ms.fbModemOn = 0;
        ms.fbModemOff = DTR_OFF;
        ulPio = sizeof ms;
        ulDio = sizeof com_err;
        rc = DosDevIOctl( hPort, IOCTL_ASYNC,
15        ASYNC_SETMODEMCTRL, &ms, ulPio, &ulPio, &com_err, ulDio,
        &ulDio );
        if (rc) return FALSE;
        else return TRUE;
    }

20    FLAG TPort::fRaisedDTR()
    {
        ULONG ulPio, ulDio;
        MODEMSTATUS ms;
25        ULONG com_err;

        ms.fbModemOn = DTR_ON;
        ms.fbModemOff = 0xFF;
        ulPio = sizeof ms;
30        ulDio = sizeof com_err;
        rc = DosDevIOctl( hPort, IOCTL_ASYNC,
        ASYNC_SETMODEMCTRL, &ms, ulPio, &ulPio, &com_err, ulDio,
        &ulDio );
        if (rc) return FALSE;
35        else return TRUE;
    }

    void _Optlink ManageThread( PVOID ); // Used internally
    by fStartManageThread().
40    void _Optlink ManageThread( PVOID ptr )
    {
        ((TPort*)ptr)->ManagePort();
    }

45    FLAG TPort::fStartManageThread()
    {
        fManThread = TRUE;
        manage_thread = _beginthread( ManageThread, 8192,
        (PVOID)this );
50        if (manage_thread == -1) return FALSE;
        else return TRUE;
    }

    void TPort::ManagePort()
55    {
```

SUBSTITUTE SHEET

```
char read_buf[32];
ULONG cbRead;

while (TRUE) {
5   rc = DosRead( hPort, read_buf, sizeof read_buf,
    &cbRead );
    if (rc) {
        // handle error here...
    }
10   else if (!fManThread) break;
    for (int i = 0; i < cbRead; i++) {
        if (log_flag) log.fPostChar( read_buf[i] );
        buffer.fPutChar( read_buf[i] );
    }
15   buffer.SignalRelease();
}

// Signal threads exit.
manage_thread = -1;
20 }

FLAG TPort::fStartCommandThread( TTHREAD CommandThread,
PVOID data )
{
25   fCmdThread = TRUE;
    command_thread = _beginthread( CommandThread, 8192,
data );
    if (command_thread == -1) return FALSE;
    else return TRUE;
30 }

#include <TStream.HPP>

#include <debug.h>
35

#include <string.h>

//*****
//*****
40 //
// TStream members.
//
TStream::TStream( UINT buf_size )
:   buf_len( buf_size ),
45   buffer( new BYTE[buf_size] ),
    iptr( buffer ),
    xptr( buffer )
{
    #ifdef DEBUG
50     memset( buffer, UNDEF_DATA, buf_len );
    #endif
}

TStream::~TStream()
55 {
```

SUBSTITUTE SHEET

```
        delete buffer;
    }

    void TStream::Reset()
    {
        iptr = xptr = buffer;
    }

    TStream& TStream::operator << ( const FLAG flag )
    {
        *(FLAG*)iptr = flag;
        return incInserter( sizeof( flag ) );
    }

    TStream& TStream::operator << ( const USHORT num )
    {
        *(USHORT*)iptr = num;
        return incInserter( sizeof( num ) );
    }

    TStream& TStream::operator << ( const ULONG num )
    {
        *(ULONG*)iptr = num;
        return incInserter( sizeof( num ) );
    }

    TStream& TStream::operator << ( const char *str )
    {
        strcpy( iptr, str );
        return incInserter( strlen( str ) + 1 );
    }

    TStream& TStream::Put( const PVOID data, UINT size )
    {
        memcpy( iptr, data, size );
        return incInserter( size );
    }

    TStream& TStream::operator >> ( FLAG &flag )
    {
        flag = *(FLAG*)xptr;
        return incExtractor( sizeof( flag ) );
    }

    TStream& TStream::operator >> ( USHORT &num )
    {
        num = *(USHORT*)xptr;
        return incExtractor( sizeof( num ) );
    }

    TStream& TStream::operator >> ( ULONG &num )
    {
        num = *(ULONG*)xptr;
        return incExtractor( sizeof( num ) );
    }
}
```

SUBSTITUTE SHEET


```
TStream& TStream::operator >> ( char *str )
{
    strcpy( str, xptr );
    return incExtractor( strlen( str ) + 1 );
5   }

TStream& TStream::Get( PVOID data, UINT size )
{
    memcpy( data, xptr, size );
    return incExtractor( size );
10  }

TStream& TStream::incExtractor( UINT n )
{
    xptr += n;
    ASSERT( xptr <= iptr );
    return *this;
15  }

TStream& TStream::incInserter( UINT n )
{
    iptr += n;
    ASSERT( iptr <= buffer + buf_len );
    return *this;
20  }
25  }

;*****
;*****
30  ;*
;* Copyright (C) 1995 Absolute Software Corporation
;*
;*****
;*****
35  NAME DBServer WINDOWCOMPAT

IMPORTS      CTIMS.fGenerateSerCTID
              CTIMS.fXlatSerCTID
40          CTIMS.fXlatCliCTID
              CTIMS.fGenerateCTCODE
              CTIMS.fConvertStrToCTCODE
              CTIMS.fConvertCTCODEToStr

45  .\TObject.obj: \
      f:\Server\TObject.CPP \
      DBServer.MAK

50  .\objects.obj: \
      f:\Server\objects.cpp \
      DBServer.MAK

55  .\MessagePipe.obj: \
      f:\Server\MessagePipe.CPP \
      DBServer.MAK
```

SUBSTITUTE SHEET

```

    .\CTMessage.obj: \
        f:\Server\CTMessage.CPP \
        DBServer.MAK

5    .\ctims.obj: \
        f:\Server\ctims.cpp \
        DBServer.MAK

    .\DBServer.obj: \
10    f:\Server\DBServer.C \

    {f:\Server;F:\Server\INCLUDE;E:\SQLLIB;E:\TOOLKT21\CPLUS\
    OS2H;E:\Tools\IBMCPP\INCLUDE;}DBServer.H \
    DBServer.MAK

15    .\TSTREAM.obj: \
        f:\Server\TSTREAM.CPP \
        DBServer.MAK

    .\TPacket.obj: \
20    f:\Server\TPacket.CPP \

    {f:\Server;M:\SRC\Include;M:\CT\Include;$(INCLUDE);}TPack
    et.HPP \
25    Server.MAK

    .\TModem.obj: \
        f:\Server\TModem.CPP \
        Server.MAK

30    .\CT_Log.obj: \
        f:\Server\CT_Log.CPP \

    {f:\Server;M:\SRC\Include;M:\CT\Include;$(INCLUDE);}CT_Lo
35    g.HPP \
        Server.MAK

    .\CT_Buffer.obj: \
        f:\Server\CT_Buffer.CPP \

40    {f:\Server;M:\SRC\Include;M:\CT\Include;$(INCLUDE);}CT_Bu
        ffer.HPP \

    {f:\Server;M:\SRC\Include;M:\CT\Include;$(INCLUDE);}serve
45    r.h \
        Server.MAK

    .\Server.obj: \
        f:\Server\Server.C \

50    {f:\Server;M:\SRC\Include;M:\CT\Include;$(INCLUDE);}CT_Tr
        ans.H \

    {f:\Server;M:\SRC\Include;M:\CT\Include;$(INCLUDE);}TPack
55    et.HPP \
```

SUBSTITUTE SHEET

Server.MAK

```
5  .\CT_Trans.obj: \
    f:\Server\CT_Trans.C \

    {f:\Server;M:\SRC\Include;M:\CT\Include;$(INCLUDE);}CT_Tr
ans.H \

10 {f:\Server;M:\SRC\Include;M:\CT\Include;$(INCLUDE);}TPack
    et.HPP \
        Server.MAK

    .\TPort.obj: \
15    f:\Server\TPort.CPP \

    {f:\Server;M:\SRC\Include;M:\CT\Include;$(INCLUDE);}TPort
.HPP \

20 {f:\Server;M:\SRC\Include;M:\CT\Include;$(INCLUDE);}CT_Bu
    ffer.HPP \

    {f:\Server;M:\SRC\Include;M:\CT\Include;$(INCLUDE);}CT_Lo
g.HPP \

25 {f:\Server;M:\SRC\Include;M:\CT\Include;$(INCLUDE);}serve
    r.h \
        Server.MAK

30 #ifndef CT_TRANS_H
    #define CT_TRANS_H

    // #include <DB_Objects.HPP>
    #include <MessagePipe.HPP>

35 #include "TPacket.HPP"

    void SntlConnect( TPort &Port, MessagePipe &Pipe,
    TConnectInfo *cnct_info );
    void SntlDisconnect( TPort &Port, TConnectInfo
40    &ConnectInfo );
    void SendDatePacket( TPort &Port, const SNTL_DATE &date
    );

    void AddDays( SNTL_DATE *next_call, int days );

45 FLAG fGetDateTime( PDATETIME );

    #endif
    #ifndef MESSAGE_H
50    #define MESSAGE_H

    /*****
    *****/
    Message.H

55
```

SUBSTITUTE SHEET

Defines all valid messages used by the Server and
ServerShell.

```
5 *****
  *****/

  // Define standard types.
  #include <os2def.h>

10  #include <time.h>

  // Definition for the Sentinel date packet.
  struct CT_DATE {
15      BYTE year;
      BYTE month;
      BYTE day;
      BYTE hour;
      BYTE minute;
20  };

  // Definition for the Sentinel serial number packet.
  struct CT_SN {
      USHORT sn[3];
      USHORT cksum;
25  CT_DATE date;
  };

  #define CND_NUM_MAXLEN      20
  #define CND_NAME_MAXLEN    20
30

  struct CALLERID_INFO {
      BYTE month;
      BYTE day;
      BYTE hour;
35  BYTE minute;
      CHAR number[CND_NUM_MAXLEN];
      CHAR name[CND_NAME_MAXLEN];
  };

40  enum TRANS_STATE {
      TRANS_OK      = 0x00,
      TRANS_BAD_CND = 0x01,
      TRANS_BAD_SN  = 0x02,
45  TRANS_BAD_DATE = 0x04
  };

  struct CT_Transaction {
      DATETIME start_time;
      CALLERID_INFO cnd;
50  CT_SN sn;
      TRANS_STATE state;
      DATETIME end_time;
  };

55  enum CT_SN_QUERY {
```

SUBSTITUTE SHEET

```

    CT_SN_OK          = 0,
    CT_SN_REDFLAG     = 1,
    CT_SN_UNKNOWN     = 2
};

5

#define CT_BUFFER_LEN 256 // Allowable
10 length of modem communications for a cycle.
#define CT_GUARD_CHAR    '!'

/* Definitions for stripped CompuTrace messages.
*****/

15
#define MAX_PHONE_NUM_LEN      16 // Max length of a
phone number string.
#define CT_SERIAL_NUM_LEN      sizeof( CT_SN ) //
20 Length of serial number packet sent by the modem.
#define MAX_ERROR_STR_LEN     32 // Max length of
an error string.

enum CTMSG_TYPE {
    CTMSG_UNDEF = 0,
25    CTMSG_CONNECT,
    CTMSG_SERIAL_NUM,
    CTMSG_ERROR_LOG,
    CTMSG_DISCONNECT
};

30
struct CT_ConnectMsg {
    time_t connect_time;
    char phone_num[MAX_PHONE_NUM_LEN];
};

35
struct CT_SerialNumMsg {
    CT_SN serial_num;
};

40
struct CT_ErrorLogMsg {
    char error_str[MAX_ERROR_STR_LEN];
};

45
struct CT_DisconnectMsg {
    time_t disconnect_time;
    char log[CT_BUFFER_LEN];
};

50
struct CTMessage {
    CTMSG_TYPE type;
    union {
        CT_ConnectMsg Connect;
        CT_SerialNumMsg SerialNum;
        CT_ErrorLogMsg ErrorLog;
55    CT_DisconnectMsg Disconnect;
    };
};
```

SUBSTITUTE SHEET

```
    } Msg;
};

5  #define MAX_CTMSG_SIZE sizeof( CTMessage )           // Max
    size of a stripped (CompuTrace) message.

/* Definitions for pipe messages.
10 *****/

// Define all valid events. The following prefixes are
used:
//      CT_           For general messages
//      CT_SER_       For server originated messages not
15 related to a transaction.
//      CT_CLI_       For client originated messages not
related to a transaction.
//      CT_SER_MSG_   For server originated messages related
to a transaction.
20 //      CT_CLI_MSG_  For client originated messages related
to a transaction.
// For more detailed information please see the proper
message structure.
enum EVENT_TYPE {
25     CT_SER_MSG_AWK,           // Server acknowledges
last received message.
     CT_SER_MSG_ERROR,        // Server has had a non-
fatal error.
30     CT_SER_MSG_FATAL,        // Server has had a
fatal error and will unconditionally terminate.
     CT_SER_MSG_MESSAGE,      // Server has a message
to be processed by the client.

     CT_SER_STOP,             // Server requests the
35 client(s) stop sending messages.
     CT_SER_START,           // Server allows the
client(s) to continue sending messages.

     CT_SER_ERROR,           // Server has had an
40 internal non-fatal error.
     CT_SER_FATAL,           // Server has had an
internal fatal error and will terminate.
     CT_SER_STRING,          // Server has a general
string to be stored.
45     CT_SER_QUIT,           // Server has requested
all clients to terminate.

     CT_CLI_MSG_AWK.         // Client acknowledges
last received message.
50     CT_CLI_MSG_ERROR,      // Client has had a non-
fatal error.
     CT_CLI_MSG_FATAL,        // Client has had a
fatal error and will unconditionally terminate.
     CT_CLI_MSG_MESSAGE      // Client has a message
55 to be processed by the server.
```

SUBSTITUTE SHEET

```
};

// Define message transfer template used to transfer a
// message through a pipe.
5 struct CT_MessageHead {
    ULONG Id; // The message id
    number.
    EVENT_TYPE type; // The event type (see
    above).
10    BYTE len; // The length the
    message data.
};

struct CT_MessageBuffer {
15    CT_MessageHead header;
    char message[MAX_CTMSG_SIZE];
};

#define MAX_MSG_SIZE sizeof( CT_MessageBuffer )
// Max size of a pipe message.

20 #endif // MESSAGE_H

#ifdef PACKET_H
25 #define PACKET_H

// Ensure byte alignment enforced!
#pragma pack( 1 ) // For C-Set++
#pragma option -a1 // For BC++

30 /* Packet Level Defines
*****/
#define STX 0x02 // Start-of-
35 text.
#define ETX 0x03 // End-of-
text.
#define EOT 0x04 // End-of-
transmission.
40 #define ENQ 0x05 // Enquiry.
#define ACK 0x06 //
Acknowledgement.
#define NAK 0x15 // Negative-
acknowledgement.

45 #define MAX_ENQ 3 // Max
number of ENQs.
#define MAX_NAK 2 // Max
number of NAKs.

50 #define MAX_TEXT_LEN 256 // Max size
of a packets TEXT.

struct PKT_HEADER {
55    BYTE stx;
    BYTE lsb_length;
```

SUBSTITUTE SHEET

```
    BYTE msb_length;
};

5    struct PKT_FOOTER {
        BYTE etx;
        BYTE lrc;
    };

10    /* Packet type definitions
    *****/

    // Text Type IDs.
    #define CTID_TEXT_TYPE                (WORD)0x0000    //
    Sentinel Subscription Number Packet.
15    #define NC_TEXT_TYPE                (WORD)0x0080    //
    Server Next Call Packet.

    struct SNTL_DATE {
20        BYTE year;
        BYTE month;
        BYTE day;
        BYTE hour;
        BYTE minute;
    };

25    struct CTID_TEXT {
        BYTE type;
        BYTE sub_type;
        WORD sn[3];
30        SNTL_DATE now_date;
    };
    #define SN_TEXT CTID_TEXT                // Old name (uses
    should be changed to CTID_TEXT).

35    struct CTID_PACKET {
        PKT_HEADER header;
        CTID_TEXT text;
        PKT_FOOTER footer;
    };
40    #define SN_PACKET CTID_PACKET            // Old name (uses
    should be changed to CTID_PACKET).

    struct NC_TEXT {
45        WORD type;
        SNTL_DATE next_call_date;
    };

    struct NC_PACKET {
50        PKT_HEADER header;
        NC_TEXT text;
        PKT_FOOTER footer;
    };

55    #pragma pack()                // Back to default.
    #pragma option -a.
```

SUBSTITUTE SHEET


```
#endif
#ifndef SERVER_H
#define SERVER_H

5   #define DEBUG          4

#include <debug.h>
#include <usertype.h>

10  //
// TConnectInfo definition.
//
#define CND_NUM_MAXLEN      20
15  #define CND_NAME_MAXLEN  20

struct CALLERID_INFO {
    BYTE month;
    BYTE day;
20  BYTE hour;
    BYTE minute;
    CHAR number[CND_NUM_MAXLEN];
    CHAR name[CND_NAME_MAXLEN];
};

25  struct TConnectInfo {
    DATETIME start_time, end_time;
    CALLERID_INFO cnd;
};
//
30  // End of TConnectInfo
//

#endif // SERVER_H
#ifndef CT_BUFFER_HPP
35  #define CT_BUFFER_HPP

#include "server.h"

#define TRUE 1
40  #define FALSE 0
#define CT_BUFFER_MAXLEN  256

class CT_Buffer {

45  char buffer[CT_BUFFER_MAXLEN];
    UINT head, tail;
    HMTX hBufSem;
    HEV hReleaseGetSem;
50  APIRET rc;

    UINT IncBufPtr( UINT ptr ) const
        { return (++ptr >= CT_BUFFER_MAXLEN) ? 0 : ptr; }

55  public:
```

SUBSTITUTE SHEET

```

    CT_Buffer();
    -CT_Buffer();

5    void Flush();

    BOOL fIsEmpty() const { return head == IncBufPtr( tail
); }
    BOOL fIsFull() const { return head == tail; }

10    void SignalRelease() { DosPostEventSem( hReleaseGetSem
); }

    BOOL fPutChar( char );
    BOOL fGetChar( char& );
15 };

    #endif
    #ifndef CT_LOG_HPP
    #define CT_LOG_HPP

20    #define TRUE 1
    #define FALSE 0

    class CT_Log {
25        char *buffer;
        UINT index, buf_len;

    public:
30        CT_Log( UINT = 4096 );
        -CT_Log();

        void Flush() { index = 0; }

35        BOOL fIsEmpty() const { return index == 0; }
        BOOL fIsFull() const { return index >= buf_len; }

        BOOL fPostChar( char );

40        BOOL fDumpLog( const char * );
    };

45    #endif
    #ifndef TCLIENT_HPP
    #define TCLIENT_HPP

50    class TClient {

        TConnectInfo ConnectInfo;
        WORD ctid[3];
55        SNTL_DATE client_date;
```

SUBSTITUTE SHEET

```

    Pipe
    public:
5
    }

10

15

    #endif // CLIENT_HPP
    #ifndef TPACKET_HPP
    #define TPACKET_HPP

20
    #include <os2def.h>
    #include "packet.h"

    #include <TPort.HPP>

25
    //*****
    // Class TPacket - Encapsulates the reception of a packet
    // for a port
30
    // TPacket::TPacket( TPort& Port ) Initializes internal
    // state.
    // Arguments:
    // TPort& Port - the port to receive the packet
35
    //
    // TRANS_STATE TPacket::rGetPacket()
    // Description:
    // Attempts to receive a packet from Port using the
40
    // protocol
    // defined in the CompuTrace Protocol Specification
    // (CTPSpec).
    //
    // Returns: The result of the attempt:
45
    // TRANS_ACK - packet successfully received as
    // defined by CTPSpec.
    // TRANS_NAK - reception aborted due to invalid
    // reception, EOT sent.
    // TRANS_ETO - ENQ timeout, no data recieved, EOT
50
    // sent.
    //
    // UINT TPacket::cbCopyText( ptr, len )
    // Arguments:
    // PVOID ptr - the buffer to copy data to.
55
    // UINT len - the maximum number of bytes to copy.
```

SUBSTITUTE SHEET

```
//
//   Description:
//   Copies text from a sucessfully received packet
5  into buffer pointed to
//   by ptr. Copies up to len bytes or the size of
the received packet
//   text (whichever is smaller). Can only be called
if rGetPacket
//   returned TRANS_ACK.
10 //
//   Returns: number of bytes copied. or 0 if packet not
successfully
//   received.
//
15 // TRANS_STATE rState() const
//   Returns: the current state of the instance.
//*****
*****
20 class TPacket {
public:

    enum TRANS_STATE {
25         TRANS_NULL,           // No
activity.
        TRANS_ACK,
        TRANS_NAK,
        TRANS_ETO };           // ETO =
Enquiry time-out.

30     TPacket( TPort& );
    TRANS_STATE rGetPacket();
    UINT CbCopyText( PVOID ptr, UINT len );

    TRANS_STATE rState() const { return state; }
35 protected:

    FLAG fSendENQ();
    FLAG fSendACK();
40    FLAG fSendNAK();
    FLAG fSendEOT();

private:

45    TPort& Port;
    int enq_count;
    int nak_count;
    USHORT text_length;
    BYTE packet_text[MAX_TEXT_LEN];
50    TRANS_STATE state;

    TRANS_STATE rReceivePacket();
};

55 #endif
```

SUBSTITUTE SHEET

```
# Created by IBM WorkFrame/2 MakeMake at 17:36:34 on
08/22/95
#
# This makefile should be run in the following directory:
5 # d:\Server
#
# The actions included in this makefile are:
# COMPILE::CLC C++
# LINK::CLC Link
10
.all: \
    .\DBServer.EXE

.SUFFIXES:

15 .SUFFIXES: .C .CPP

.CPP.obj:
    @echo WF::COMPILE::CLC C++
20    icc.exe /Tl- /Xi /ID:\Server\INCLUDE /IE:\SQLLIB
    /IE:\TOOLKT21\CPLUS\OS2H /IE:\Tools\IBMCPP\INCLUDE
    /DDEBUG=4 /Tdp /Q /Wall /Fi /Ti /Gm /G5 /Tm /C %s

.C.obj:
25    @echo WF::COMPILE::CLC C++
    icc.exe /Tl- /Xi /ID:\Server\INCLUDE /IE:\SQLLIB
    /IE:\TOOLKT21\CPLUS\OS2H /IE:\Tools\IBMCPP\INCLUDE
    /DDEBUG=4 /Tdp /Q /Wall /Fi /Ti /Gm /G5 /Tm /C %s

30 .\DBServer.EXE: \
    .\TObject.obj \
    .\TSTREAM.obj \
    .\DBServer.obj \
    .\ctims.obj \
35    .\CTMessage.obj \
    .\MessagePipe.obj \
    .\objects.obj \
    {$(LIB)}DB_Objects.LIB \
    {$(LIB)}SQL_DYN.LIB \
40    {$(LIB)}DBServer.DEF \
    DBServer.MAK
    @echo WF::LINK::CLC Link
    icc.exe @<<
45 /Tl- /Xi
    /ID:\Server\INCLUDE
    /IE:\SQLLIB
    /IE:\TOOLKT21\CPLUS\OS2H
    /IE:\Tools\IBMCPP\INCLUDE
    /DDEBUG=4
50 /Tdp /Q
    /Wall
    /Fi
    /Ti /Gm /G5 /Tm
    /B" /de"
55 /FeDBServer.EXE
```

SUBSTITUTE SHEET

```
DB_Objects.LIB
SQL_DYN.LIB
DBServer.DEF
5  .\TObject.obj
   .\TSTREAM.obj
   .\DBServer.obj
   .\ctims.obj
   .\CTMessage.obj
10  .\MessagePipe.obj
   .\objects.obj
<<

15  !include DBServer.Dep
   # Created by IBM WorkFrame/2 MakeMake at 10:20:11 on
   05/30/95
   #
   # This makefile should be run in the following directory:
20  #   d:\Server
   #
   # The actions included in this makefile are:
   #   COMPILE::CLC C++
   #   LINK::CLC Link

25  .all: \
     .\Server.EXE

   .SUFFIXES:

30  .SUFFIXES: .C .CPP

   .CPP.obj:
       @echo WF::COMPILE::CLC C++
       icc.exe /Tl- /ID:\Server\Include /IM:\CT\Include
35  /Tdp /Q /Wall /Fi /Si /Ti /O /Gm /G5 /Tm /C %s

   .C.obj:
       @echo WF::COMPILE::CLC C++
       icc.exe /Tl- /ID:\Server\Include /IM:\CT\Include
40  /Tdp /Q /Wall /Fi /Si /Ti /O /Gm /G5 /Tm /C %s

   .\Server.EXE: \
       .\TPacket.obj \
       .\TPort.obj \
45  .\CT_Trans.obj \
       .\Server.obj \
       .\CT_Buffer.obj \
       .\CT_Log.obj \
       .\TModem.obj \
50  {$(LIB)}CTIMS.LIB \
       {$(LIB)}MessagePipe.LIB \
       Server.MAK
       @echo WF::LINK::CLC Link
       icc.exe @<<
55  /Tl-
```

SUBSTITUTE SHEET

```

/ID:\Server\Include
/IM:\CT\Include
/Tdp /Q
/Wall
5  /Fi /Si
/Ti /O /Gm /G5 /Tm
/B" /de"
/FeServer.EXE
CTIMS.LIB
10 MessagePipe.LIB
.\TPacket.obj
.\TPort.obj
.\CT_Trans.obj
.\Server.obj
15 .\CT_Buffer.obj
.\CT_Log.obj
.\TModem.obj
<<

20 !include Server.Dep
#ifndef CTID_H
#define CTID_H

25 /*** MOVE TO USERTYPE ***/
/* #define LOWORD( x ) ((WORD)((DWORD)(x)))
   #define HIWORD( x ) ((WORD)((x) >> 16))*/
/*****/

30 #define CTCODE_STR_LEN      10

typedef WORD *CTCODE;

extern "C" {
35 //
// fGenerateSerCTID - Creates a new valid Server CTID
// value.
//
40 FLAG APIENTRY fGenerateSerCTID( ULONG &ctid );

//
// fXlatSerCTID - Translates a ServerCTID to a
// ClientCTID.
//
45 FLAG APIENTRY fXlatSerCTID( ULONG &cli_ctid, ULONG
ser_ctid );

//
// fXlatCliCTID - Translates a ClientCTID to a
50 ServerCTID.
//
FLAG APIENTRY fXlatCliCTID( ULONG &ser_ctid, ULONG
cli_ctid );

55 //
```

SUBSTITUTE SHEET

```
// fGenerateCTCODE - Creates a 48 bit CTCODE from a valid
Client CTID.
//
5 FLAG APIENTRY fGenerateCTCODE( CTCODE ctcode, ULONG
cli_ctid );

//
// fConvertStrToCTCODE - Converts a string to CTCODE.
// Arguments - WORD *ctcode: an array of 3 WORDS to be
10 set to the 48 bit
// binary representation of the input
string.
// STRING str: the input string of size
CTID_STR_LEN.
15 //
FLAG APIENTRY fConvertStrToCTCODE( CTCODE ctcode, STRING
str );

//
20 // fConvertCTCODEToStr - Converts a CTCODE number to a
string.
// Arguments - char *str: the output string of size
CTID_STR_LEN.
// WORD *ctcode: the input array of 3
25 WORDS.
//
FLAG APIENTRY fConvertCTCODEToStr( char *str, const
CTCODE ctcode );
30 }; // end extern "C"

#endif // CTID_H
#ifndef CTIMS_H
35 #define CTIMS_H

#include <usertype.h>

#ifdef __cplusplus
40 extern "C" {
#endif

#define CALLERID_SIZE 21
#define CTSTATUS_SIZE 9
45 #define CTORGNUM_SIZE 9
#define CTTS_SIZE 27

typedef struct {
long CTID;
50 char LicStatus[CTSTATUS_SIZE];
long PeriodDays;
long PeriodMinutes;
char StolenFlag;
long SpecialProcess;
char LastCallTS_N[CTTS_SIZE];
55 short IsNull_LastCallTS;
```

SUBSTITUTE SHEET


```

    char NextCallTS_N[CTTS_SIZE];
    short IsNull_NextCallTS;
    char NextCallClientTS_N[CTTS_SIZE];
    short IsNull_NextCallClientTS;
5    char Orgnum_N[CTORGNUM_SIZE];
    short IsNull_Orgnum;
    char ProductType[CTSTATUS_SIZE];

10    } _CTlicense;

    typedef struct {
        long CTID;
        char Status[CTSTATUS_SIZE];
        char LastCallTS_N[CTTS_SIZE];
15        char NextCallTS_N[CTTS_SIZE];
        char NextCallClientTS_N[CTTS_SIZE];

    } _CTupdateLicenseStatus;

20    typedef struct {
        long CTID;
        char ServerTS[CTTS_SIZE];
        char ClientTS[CTTS_SIZE];
        char TelcoTS_N[CTTS_SIZE];
25        short IsNull_TelcoTS;
        short DurationSec_N;
        short IsNull_DurationSec;
        char CallerID_N[CALLERID_SIZE];
        short IsNull_CallerID;
30        short LineNum;
        char LogFlag;
        char EnvironmentID[CTSTATUS_SIZE];
        short ErrorCnt;

35    } _CTmonitorEvent;

    /* CTIMS.SQC */
    FLAG _fQueryLicense( _CTlicense*, ULONG CTID );
    FLAG _fUpdateLicenseStatus( const _CTupdateLicenseStatus*
40    );

    FLAG _fInsertIntoMonitorEvent      ( const
        _CTmonitorEvent* );
    FLAG _fInsertIntoMonitorEventStolen ( const
        _CTmonitorEvent* );
45    FLAG _fInsertIntoMonitorEventExpired( const
        _CTmonitorEvent* );

    /* Index.SQC */
50    long _lLastSQLCODE();
    FLAG _fGetNextTableIndex( ULONG *index, ULONG *count,
        STRING ViewName );

    /* ORG01.SQC */
```

SUBSTITUTE SHEET

```
FLAG_fMayRemoveCustomer( STRING orgnum );           //
Checks if a customer may be removed.
FLAG_fDbArchiveCustomer( STRING orgnum );           //
Archives a customer.
5 FLAG_fDbDeleteCustomer ( STRING orgnum );          //
Deletes a customer and all associated data.
FLAG_fDbDeleteOrg( STRING orgnum );                //
Deletes an org and all associated data.
10 FLAG_fIsACustomer( STRING orgnum, FLAG exclusive ); //
Determines whether an org is a customer.

#ifdef __cplusplus
}
#endif
15

#endif // CTIMS_H
#ifndef DB_H
#define DB_H

20 #include "DB_Structs.H"

#ifdef __cplusplus
extern "C" {
25 #endif

FLAG fInitDB();
FLAG fConnectDB( PCSZ db_str );

30 ULONG ulGetSQLCode();

void CommitWork();
void RollbackWork();

35 #ifdef __cplusplus
}
#endif

#endif // DB_H
#ifndef DBSERVER_H
#define DBSERVER_H

40 #define SHIP 0
#define DEBUG 4

45 #include <debug.h>
#include <usertype.h>

#endif // SERVER_H
#ifndef DB_STRUCTS_H
#define DB_STRUCTS_H
50

#ifdef __cplusplus
extern "C" {
55 #endif
```

SUBSTITUTE SHEET

```
#pragma pack( 1 )

typedef struct _TimeStampStruct {
5   char year[4];
    char dash1;
    char month[2];
    char dash2;
    char day[2];
10   char dash3;
    char hour[2];
    char dot1;
    char minute[2];
    char dot2;
15   char second[2];
    char dot3;
    char microsec[6];
} TimeStampStruct;

typedef struct _MonitorEventStruct {
20   ULONG CompuTraceID;
    TimeStampStruct ServerTS;
    TimeStampStruct PropertyTS;
    TimeStampStruct TelcoTS;
    char CallerID[20];
25   SHORT CallSeconds;
    char EnvID[8];
} MonitorEventStruct;

#pragma pack()

30  #ifdef __cplusplus
    }
    #endif

35  #endif // DB_STRUCTS_H
    #ifndef DEBUG_H
    #define DEBUG_H
    /*******
    *****/
40  //
    // DEBUG_H - sets the debug level of the code.
    //      #define SHIP = 1 and #undef DEBUG for ship code.
    //
    //      #define SHIP = 0 and DEBUG is defined for debug
45  code.
    //      DEBUG = 1 - beta level, PRECONDITION active.
    //      DEBUG = 2 - alpha level, adds CONDITION.
    //      DEBUG = 3 - pre-alpha level, adds CHECK.
    //      DEBUG = 4 - sanity check level, adds
50  SANITYCHECK.
    //
    /*******
    *****/

55  #ifdef DEBUG
```

SUBSTITUTE SHEET

```

    #define ASSERT( x )
    #define NOTIMPLEMENTED
    implemented error */ )
    assert( x )
    assert( 0 /* Not

5   #else

    #define NDEBUG
    assert.h
    // Disables debugging in
    #define ASSERT( x )
    #define NOTIMPLEMENTED
    (void)0
    (void)0

10   #endif // DEBUG

    #include <assert.h>

15   #if DEBUG >= 1
    #define PRECONDITION( x )
    #else
    #define PRECONDITION( x )
    (void)0
20   #endif

    #if DEBUG >= 2
    #define CONDITION( x )
    #else
    #define CONDITION( x )
    (void)0
25   #endif

    #if DEBUG >= 3
    #define CHECK( x )
    #else
    #define CHECK( x )
    (void)0
30   #endif

    #if DEBUG >= 4
    #define SANITYCHECK( x )
    #else
    #define SANITYCHECK( x )
    (void)0
35   #endif

40   #define UNDEF_DATA
    Used to show unallocated memory.
    #define JUNK
    UNDEF_DATA
    #define UNINIT_DATA
    Used to show uninitialized data.
    0xDD
    //
45   #endif // DEBUG_H
    #ifndef USERTYPE_H
    #define USERTYPE_H

50   #ifdef __OS2__
    #include <os2def.h>
    #endif

    #ifndef __CSET
    #define __Optlink
55
```

SUBSTITUTE SHEET

```
#endif

// Standard typedef's for Absolute Software.

5  #define MAX( x, y )      ((x) > (y) ? (x) : (y))
   #define MIN( x, y )      ((x) < (y) ? (x) : (y))

   #ifndef NULL
10  #define NULL 0
   #endif

   #define TRUE            1
   #define FALSE           0

15  typedef unsigned char FLAG;
   typedef unsigned char BYTE;

   typedef unsigned char UCHAR;
   typedef unsigned short USHORT;
20  typedef unsigned int  UINT;
   typedef unsigned long ULONG;

   #ifndef _Windows
   typedef unsigned short WORD;
25  typedef unsigned long DWORD;
   #endif
   typedef const char* STRING;

   typedef const void* PCVOID;

30  #ifdef __OS2__
   typedef void (* _Optlink TTHREAD)( PVOID );
   #endif

35  #ifdef __cplusplus
   template <class T1, class T2> FLAG operator == ( T1 c1,
   T2 c2 )
   {
40     return FLAG( c1 == c2 );
   }

   template <class T1, class T2> FLAG operator != ( T1 c1,
   T2 c2 )
   {
45     return FLAG( c1 != c2 );
   }
   #endif // __cplusplus

   #endif // USERTYPE_H
50  #ifndef CTIMS_HPP
   #define CTIMS_HPP

   #include <iostream.h>
   #include <string.h>

55
```

SUBSTITUTE SHEET

```
#define INCL_DOSDATETIME
#include <os2.h>

#include <debug.h>
5 //include <packet.h>

#include <Objects.HPP>
#include <CTIMS.H>

10 #pragma pack( 1 )
CTStatus, CTOrgNum. // Needed for

#define CT_TOK_SIZE 8

15 #define UNUSED_TOK "UNUSED "
#define NOTEST_TOK "NOTEST "
#define ACTIVE_TOK "ACTIVE "
#define EXPIRED_TOK "EXPIRED "
#define UNDEFINED_TOK " "

20 //define Y_TOK "Y"
//define N_TOK "N"
//define UNDEF_FLAG_TOK " "

25 #define ORGNUM_SIZE 8
#define ORGNUM_PREFIX_SIZE 4

//BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
30 BBBBBBBBBBBBBBBBBBBBBBBBBB
//
// CTIMS General types.
//
// The following types are general types used in CTIMS.
// They are not specific
35 // to a single implementation:
//
// TFlags - used for boolean fields such as StolenFlag
// and InsuredFlag.
// TTimestamp - used to represent timestamps with
40 millisecond resolution.
// TString - represents a string of characters.
//
//
//BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
45 BBBBBBBBBBBBBBBBBBBBBBBB

//fffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff
//
50 // CTIMS Flag
//
//fffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff
//
//*****
55 class CTFlag : public TFlag {
```

SUBSTITUTE SHEET

public:

```
CTFlag() : TFlag() {}  
CTFlag( FLAG flag ) : TFlag( flag ) {}  
5  };  
    *****/  
    typedef TFlag CTFlag;
```

```
10  //fffffffffffffffffffffffffffffffffffffffffffff  
    ffffffffffffffffffffff  
    //  
    // CTIMS Timestamp  
    //  
15  //fffffffffffffffffffffffffffffffffffffffffffff  
    ffffffffffffffffffffff  
    /*****  
    class CTTimestamp : public TTimestamp {  
    public:
```

```
20  CTTimestamp() : TTimestamp() {}  
    CTTimestamp( USHORT yr,  
                  UCHAR mo,  
                  UCHAR dy,  
25  UCHAR hr = 0,  
                  UCHAR mn = 0,  
                  UCHAR sc = 0,  
                  USHORT ms = 0 ) : TTimestamp( yr, mo, dy,  
30  hr, mn, sc, ms ) {}  
    };
```

```
    *****/  
    typedef TTimestamp CTTimestamp;
```

```
35  // UNDER CONSTRUCTION!!!  
    //fffffffffffffffffffffffffffffffffffffffffffff  
    ffffffffffffffffffffff  
    //  
    // CTIMS Status  
    //  
40  //fffffffffffffffffffffffffffffffffffffffffffff  
    ffffffffffffffffffffff  
    class CTStatus : public virtual TNull {  
    public:
```

```
45  CTStatus();  
    CTStatus( STRING str );  
  
    operator STRING() const { return value; }  
50  friend ostream& operator << ( ostream&, const  
    CTStatus& );  
  
    friend TStream& operator << ( TStream&, const  
55  CTStatus& );
```

SUBSTITUTE SHEET

SUBSTITUTE SHEET


```
    }

    operator STRING() const { useAsValue(); return value;
5  }

    friend ostream& operator << ( ostream &os, const
    CTCallerID &id )
    {
10      if (id.fIsNull()) return os << "NULL";
        else return os.write( id.value,
        sizeof( id.value ) );
    }

    friend TStream& operator << ( TStream&, const
15  CTCallerID& );
    friend TStream& operator >> ( TStream&,
    CTCallerID& );

private:
20  char value[CALLERID_SIZE];
    };

    //fffffffffffffffffffffffffffffffffffffffffffffffffffff
    ffffffffffffffffffffffffff
25  //
    // CTIMS License.Status
    //
    //fffffffffffffffffffffffffffffffffffffffffffffffffffff
    ffffffffffffffffffffffffff
30  class CTLicStatus : public virtual TNull {
public:

    enum VALUE {
35      UNUSED = 0,
        NOTEST = 1,
        ACTIVE = 2,
        EXPIRED = 3
40      };

    CTLicStatus() : TNull( TRUE ) {}
    CTLicStatus( VALUE val ) : TNull( FALSE ), value( val
    ) {}
    CTLicStatus( STRING str );
45  CTLicStatus& operator = ( VALUE newval )
    {
        setNotNull();
        value = newval;
50      return *this;
    }
    CTLicStatus& operator = ( STRING );

    operator STRING() const { useAsValue(); return
55  STR_SET[value]; }
```

SUBSTITUTE SHEET

```
operator VALUE() const { useAsValue(); return value; }

FLAG operator == ( const CTLicStatus &status ) const
{ useAsValue(); return FLAG( value == (VALUE)status
5  ); }
FLAG operator == ( VALUE val ) const
{ useAsValue(); return FLAG( value == val ); }

friend ostream& operator << ( ostream &os, const
10 CTLicStatus &lic )
{
    if (lic.fIsNull()) return os << "NULL";
    else return os << STRING( lic );
15 }

friend TStream& operator << ( TStream&, const
CTLicStatus& );
friend TStream& operator >> ( TStream&,
20 CTLicStatus& );

private:
    static const char STR_SET[][CT_TOK_SIZE+1];
    VALUE value;
25 };

//fffffffffffffffffffffffffffffffffffffffffffff
//
// CTIMS Organization Number.
30 //
//fffffffffffffffffffffffffffffffffffffffffffff
//
class CTOrgnum : public virtual TNull {
35 public:

    CTOrgnum() : TNull( TRUE ) {}

    FLAG fSetPrefix( STRING );
40 FLAG fSetIndex( UINT );

    FLAG fGetPrefix( char * ) const;
    FLAG fGetIndex( UINT &i ) const;

45 FLAG fGeneratePrefix( STRING org_name );

    operator STRING() const;

    CTOrgnum& operator = ( STRING str )
50 {
    setNotNull();
    strncpy( value, str, sizeof( value ) );
    return *this;
55 }
```

SUBSTITUTE SHEET

```
friend ostream& operator << ( ostream &os, const
CTOrgnum &lic )
{
    if (lic.isNull()) return os << "NULL";
5      else          return os.write( lic.value, sizeof(
lic.value ) );
}

10      friend TStream& operator << ( TStream&, const
CTOrgnum& );
      friend TStream& operator >> ( TStream&,
CTOrgnum& );

private:
15      char value[ORGNUM_SIZE];
};

20      //BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
BBBBBBBBBBBBBBBBBBBBBBBB
//
// CTIMS Records.
//
25      // The following types represent records stored in CTIMS.
//
//
//BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
BBBBBBBBBBBBBBBBBBBBBBBB
30
//ffffffffffffffffffffffffffffffffffffffffffffffffffffffffff
ffffffffffffffffffff
//
// CTIMS MonitorEvent
35      //
//ffffffffffffffffffffffffffffffffffffffffffffffffffffffffff
ffffffffffffffffffff
struct CTMonitorEvent
{
40      ULONG          CTID;
      CTTimestamp ServerTS;
      CTTimestamp ClientTS;
      CTTimestamp TelcoTS_n;
      USHORT          DurationSec_n;
45      CTCallerID CallerID_n;
      USHORT          LineNum;
      CTFlag          LogFlag;
      CTStatus        EnvironmentID;
      USHORT          ErrorCnt;
50
      friend ostream& operator << ( ostream&, const
CTMonitorEvent& );

      friend TStream& operator << ( TStream&, const
CTMonitorEvent& );
55
```

SUBSTITUTE SHEET

```
        friend TStream& operator >> ( TStream&,
        CTMonitorEvent& );
    };

5    #pragma pack()

    #endif // CTIMS_HPP
    #ifndef CTMESSAGE_HPP
10    #define CTMESSAGE_HPP

    #include <stddef.h>

    #include <TStream.HPP>
    #include <CTIMS.HPP>
15

    /*******
    *****/
    //
    //
20    /*******
    *****/

    // CT Message Type Enum.
    enum CT_MSG_TYPE {
25        QUERY_CTID STATUS,
        CTID STATUS RESULT,
        STORE_MONITOR_EVENT,
        STORE_RESULT,
        CLI_QUIT
30    };

    inline TStream& operator << ( TStream &buf, const
    CT_MSG_TYPE type )
    {
35        return buf << USHORT( type );
    }

    inline TStream& operator >> ( TStream &buf, CT_MSG_TYPE
    &type )
40    {
        USHORT num;
        buf >> num;
        type = CT_MSG_TYPE( num );
        return buf;
45    }

    //
    // Header for all CT Messages.
    //
50    class CTMessageHeader {
    public:

        CTMessageHeader() {}
        CTMessageHeader( ULONG id, CT_MSG_TYPE type, USHORT
55    len )
```

SUBSTITUTE SHEET

```
        : ID( id ), Type( type ), Len( len )
    {}

    CT_MSG_TYPE eType() const { return Type; }

5    friend TStream& operator << ( TStream&, const
    CTMessageHeader& );
    friend TStream& operator >> ( TStream&,
    CTMessageHeader& );

10    protected:
        ULONG ID;                // The message id
        number.
        CT_MSG_TYPE Type;        // The event type (see
15    above).
        USHORT Len;              // The length the
        message data.
    };

20    //
    // Template for message types.
    //
    template < class TText, CT_MSG_TYPE type >
    class CTMessage : public CTMessageHeader, public TText {
25    public:

        CTMessage()
        : CTMessageHeader( 0, type, sizeof( *this ) )
30    {}

        CTMessage( const CTMessageHeader &Header )
        : CTMessageHeader( Header )
        {
35            ASSERT( Type == type );
        }

        friend TStream& operator << ( TStream &buf, const
    CTMessage< TText, type > &msg )
40    {
        return buf << *(const CTMessageHeader*)&msg <<
        *(const TText*)&msg;
    }

        friend TStream& operator >> ( TStream &buf, CTMessage<
    TText, type > &msg )
45    {
        return buf >> *(CTMessageHeader*)&msg >>
        *(TText*)&msg;
    }
50    };

    /*****
    // Doesn't seem to work in OS/2 BC++.
    template < class TText, CT_MSG_TYPE type >
```

SUBSTITUTE SHEET

```
TStream& operator << ( TStream &buf, const CTMessage<
TText, type > &msg )
{
    return buf << *(const CTMessageHeader*)&msg << *(const
5 TText*)&msg;
}

template < class TText, CT_MSG_TYPE type >
TStream& operator >> ( TStream &buf, CTMessage< TText,
10 type > &msg )
{
    return buf >> *(CTMessageHeader*)&msg >>
*(TText*)&msg;
}
15 *****/

//
// CT Message structures.
//
20 struct QueryCTIDStatus {
    ULONG CTID;

    friend TStream& operator << ( TStream&, const
QueryCTIDStatus& );
25 friend TStream& operator >> ( TStream&,
QueryCTIDStatus& );
};

inline TStream& operator << ( TStream &buf, const
QueryCTIDStatus &rec )
{
    return buf << rec.CTID;
}

30 inline TStream& operator >> ( TStream &buf,
QueryCTIDStatus &rec )
{
    return buf >> rec.CTID;
}
35

40 struct CTIDStatusResult {
    FLAG      QueryResult;

    ULONG      CTID;
45 CTLicStatus Status;
    ULONG      PeriodDays;
    ULONG      PeriodMinutes;
    CTFlag      StolenFlag;
    ULONG      SpecialProcess;
50 CTTimestamp LastCallTS_n;
    CTTimestamp NextCallTS_n;
    CTTimestamp NextCallClientTS_n;
    CTOrgnum    Orgnum_n;
    CTStatus    ProductType;
55
```

SUBSTITUTE SHEET

```
    friend TStream& operator << ( TStream&, const
CTIDStatusResult& );
    friend TStream& operator >> ( TStream&,
CTIDStatusResult& );
5    };

    inline TStream& operator << ( TStream &buf, const
CTIDStatusResult &rec )
    {
10        return buf << rec.QueryResult
            << rec.CTID
            << rec.Status
            << rec.PeriodDays
            << rec.PeriodMinutes
15            << rec.StolenFlag
            << rec.SpecialProcess
            << rec.LastCallTS_n
            << rec.NextCallTS_n
            << rec.NextCallClientTS_n
20            << rec.Orgnum_n
            << rec.ProductType;
    }

    inline TStream& operator >> ( TStream &buf,
CTIDStatusResult &rec )
25    {
        return buf >> rec.QueryResult
            >> rec.CTID
            >> rec.Status
30            >> rec.PeriodDays
            >> rec.PeriodMinutes
            >> rec.StolenFlag
            >> rec.SpecialProcess
            >> rec.LastCallTS_n
35            >> rec.NextCallTS_n
            >> rec.NextCallClientTS_n
            >> rec.Orgnum_n
            >> rec.ProductType;
    }
40

    struct StoreMonitorEvent : public CTMonitorEvent {

        // Control.
        FLAG StoreAsStolen;
45        FLAG StoreAsExpire;

        // Data.
        CTLicStatus    LicenseStatus;
        CTTimestamp    NextCallTS_n;
50        CTTimestamp    NextCallClientTS_n;

        friend TStream& operator << ( TStream&, const
StoreMonitorEvent& );
        friend TStream& operator >> ( TStream&,
StoreMonitorEvent& );
55    }
```

SUBSTITUTE SHEET

```
};

inline TStream& operator << ( TStream &buf, const
5 StoreMonitorEvent &rec )
{
    return buf << rec.StoreAsStolen
        << rec.StoreAsExpire
        << rec.LicenseStatus
        << rec.NextCallTS_n
10 << rec.NextCallClientTS_n
        << (const CTMonitorEvent&)rec;
}

inline TStream& operator >> ( TStream &buf,
15 StoreMonitorEvent &rec )
{
    return buf >> rec.StoreAsStolen
        >> rec.StoreAsExpire
        >> rec.LicenseStatus
        >> rec.NextCallTS_n
20 >> rec.NextCallClientTS_n
        >> (CTMonitorEvent&)rec;
}

25 struct StoreResult {
    FLAG Result;

    friend TStream& operator << ( TStream&, const
30 StoreResult& );
    friend TStream& operator >> ( TStream&,
    StoreResult& );
};

35 inline TStream& operator << ( TStream &buf, const
    StoreResult &rec )
{
    return buf << rec.Result;
}

40 inline TStream& operator >> ( TStream &buf, StoreResult
    &rec )
{
    return buf >> rec.Result;
}

45 struct CliQuit {
    friend TStream& operator << ( TStream &buf, const
    CliQuit& ) { return buf; }
    friend TStream& operator >> ( TStream &buf,
50 CliQuit& ) { return buf; }
};

typedef CTMessage< QueryCTIDStatus, QUERY_CTID_STATUS >
QueryCTIDStatusMsg;
```

SUBSTITUTE SHEET


```
typedef CTMessage< CTIDStatusResult, CTID_STATUS_RESULT >
CTIDStatusResultMsg;

typedef CTMessage< StoreMonitorEvent, STORE_MONITOREVENT
5 > StoreMonitorEventMsg;
typedef CTMessage< StoreResult, STORE_RESULT >
StoreResultMsg;

typedef CTMessage< CliQuit, CLI_QUIT > CliQuitMsg;
10

#endif // CTMESSAGE_HPP
#ifndef DB_OBJECTS_HPP
#define DB_OBJECTS_HPP

15 #include <DB.H>

#define DB_NULL -1
#define DB_NOT_NULL 0
#define DB_ISNULL( n ) (FLAG( (n) < 0 ))

20 class DataBase {

    PCSZ name;

25 public:

    DataBase() { fInitDB(); }
    DataBase( PCSZ db_name ) : name( db_name ) {
        fInitDB(); }
30

    void SetName( PCSZ str ) { name = str; }
    FLAG fConnect() { return fConnectDB( name ); }

    ULONG ulSQLCode() const { return ulGetSQLCode(); }
35

    void Commit() { CommitWork(); }
    void Rollback() { RollbackWork(); }
};

40 #endif // DB_OBJECTS_HPP
#ifndef MESSAGEPIPE_HPP
#define MESSAGEPIPE_HPP

45 #include <debug.h>
#include <usertype.h>
#include <TStream.HPP>

//*****
//*****
50 // MsgPipeFactory - Factory to create MessagePipe
instances.
// Each MessagePipe instance represents a connection
between a
// client and the server.
55 //
```

SUBSTITUTE SHEET

```
// *** PUBLIC INTERFACE ***
//
// FLAG fCreatePipe( MessagePipe* *pipe )
//   Description:
5 //   Creates a MessagePipe instance and returns a
//   pointer to it (via pipe).
//   Returns:
//   TRUE if the operation is successful.
//   FALSE if the operation fails.
10 //
// FLAG fDestroyPipe( MessagePipe *pipe )
//   Description:
//   Destroys the MessagePipe instance pointed to by
//   pipe.
15 //   Returns:
//   TRUE if the operation is successful.
//   FALSE if the operation fails.
//
// *** PROTECTED INTERFACE ***
20 //
// virtual void InitPipe( MessagePipe *pipe )
// virtual void DeinitPipe( MessagePipe *pipe )
//   Description:
//   Called by the constructor or destructor of
25 MessagePipe respectively.
//   Manages any internal work needed to support an
//   MessagePipe instance.
//
// virtual FLAG fOpenPipe( MessagePipe* )
30 // virtual FLAG fClosePipe( MessagePipe* )
//   Description:
//   Called by MessagePipe::fOpenPipe and
//   MessagePipe::fClosePipe. This
//   in turn allocates/deallocated a pipe using the
35 needed OS API calls.
//
//*****
//*****
40 class MsgPipeFactory {
    friend class MessagePipe;

public:
45     MsgPipeFactory( UINT msg_len )
        : max_msg_len( msg_len ),
          rc( 0 )
    {}
    virtual ~MsgPipeFactory() {}

50     virtual FLAG fCreatePipe( MessagePipe*& ) = 0;
    virtual FLAG fDestroyPipe( MessagePipe* ) = 0;

    UINT uMaxMsgLen() const { return max_msg_len; }
55     APIRET rcDosErrorCode() const { return rc; }
```

SUBSTITUTE SHEET

```
protected:

    virtual void InitPipe( MessagePipe* ) {}
    virtual void DeinitPipe( MessagePipe* ) {}

5    virtual FLAG fOpenPipe( MessagePipe* ) = 0;
    virtual FLAG fClosePipe( MessagePipe* ) = 0;

    APIRET rc;

10    private:

        UINT max_msg_len;
    };

15    //*****
    // SvrMsgPipeFactory - Factory to create MessagePipe
    instances from the
    //      Server process.
    //
    // See MsgPipeFactory.
    //
    //*****
25    class SvrMsgPipeFactory : public MsgPipeFactory {

    public:

30        SvrMsgPipeFactory( PCSZ pipe_name, UINT max_msg_size,
        UINT max_msg_num );
        ~SvrMsgPipeFactory() {}

        FLAG fCreatePipe( MessagePipe*& );
35        FLAG fDestroyPipe( MessagePipe* );

    protected:

        // void InitPipe( MessagePipe* );
40        // void DeinitPipe( MessagePipe* );

        FLAG fOpenPipe( MessagePipe* );
        FLAG fClosePipe( MessagePipe* );

45    private:

        PCSZ pipe_name;
        UINT pipe_len;
    };

50    //*****
    // CltMsgPipeFactory - Factory to create MessagePipe
    instances from the
55    //      Client process.
```

SUBSTITUTE SHEET

```
//
// See MsgPipeFactory.
//
5 //*****
//*****
class CltMsgPipeFactory : public MsgPipeFactory {
public:
10     CltMsgPipeFactory( PCSZ pipe_name, UINT max_msg_size
);
    -CltMsgPipeFactory() {}

    FLAG fCreatePipe( MessagePipe*& );
15     FLAG fDestroyPipe( MessagePipe* );

protected:

20     // void InitPipe( MessagePipe* );
    // void DeinitPipe( MessagePipe* );

    FLAG fOpenPipe( MessagePipe* );
    FLAG fClosePipe( MessagePipe* );

25 private:

    PCSZ pipe_name;
};

30 //*****
//*****
// Class MessagePipe - Implements a message pipe
connection between the client
// and the server. This same class is used for both
35 the client and the
// server sides. MsgPipeFactory is used to hide the
connection differences.
//
// FLAG fOpenPipe()
40 // FLAG fClosePipe()
// Description:
// Called to open/close a valid connection between
the client and the
// server. fOpenPipe must be called before any
45 data can be transferred.
//
// FLAG fSendMessage( PCVOID msg, ULONG msg_len )
// Description:
// Sends msg[msg_len] through the pipe as raw data.
50 // Returns: TRUE = success; FALSE = failure.
//
// FLAG fGetMessage( PVOID msg, PULONG msg_len )
// Description:
// Receives up to msg_len byte into msg. Does not
55 return until a message
```

SUBSTITUTE SHEET

```
//      is recieved.
//      Returns: TRUE = success; FALSE = failure.
//
// FLAG fTransact( PCVOID out_msg, ULONG out_msg_len,
5 PVOID in_msg,
//      PULONG in_msg_len )
//      Description:
//      Sends out_msg and then receives in_msg. Does
not return until a
10 //      message has been received.
//      Returns: TRUE = success; FALSE = failure.
//
// PIPE_STATE eState()
//      Returns:
15 //      The current state of the pipe:
//      DISCONNECTED - the pipe is not connected to
another process.
//      LISTENING - the pipe is waiting for the two
sides to connect.
20 //      CONNECTED - the pipe is connected; data
transfer is allowed.
//      CLOSING - pipe is waiting for one side to
acknowledge closure.
//
25 // UINT uMaxMsgLen() const
//      Returns:
//      The maximum message length that can be sent or
received.
//
30 // APIRET rcDosErrorCode() const
//      Returns:
//      The OS API return code of the last API
operation. Commonly used
//      to determine the type of error once a FALSE has
35 //      been returned by
//      one of the member functions above.
//
//*****
*****
40 class MessageBuffer;          // Forward declaration.

class MessagePipe {

    friend class SvrMsgPipeFactory;
45     friend class CltMsgPipeFactory;

    MessagePipe( MsgPipeFactory* );
    ~MessagePipe();

50 public:

    // Pipe state enum. Fixed numbers are set to match API
state (see implementation)!
    enum PIPE_STATE {
55         DISCONNECTED = 1,
```

SUBSTITUTE SHEET

```

    LISTENING = 2,
    CONNECTED = 3,
    CLOSING = 4
};

5    FLAG fOpenPipe();
    FLAG fClosePipe();

    FLAG fSendMessage( PCVOID msg, ULONG msg_len );
10    FLAG fGetMessage( PVOID msg, PULONG msg_len );
    FLAG fTransact( PCVOID out_msg, ULONG out_msg_len,
    PVOID in_msg, PULONG in_msg_len );

    FLAG fSendMessage( TStream& );
15    FLAG fGetMessage( TStream& );
    FLAG fTransact( TStream &out, TStream &in );

    PIPE_STATE eState();
    UINT uMaxMsgLen() const { return factory-
20    >uMaxMsgLen(); }
    APIRET rcDosErrorCode() const { return rc; }

protected:

25    void SetHandle( HPIPE h ) { hPipe = h; }
    HPIPE GetHandle() const { return hPipe; }

private:

30    MsgPipeFactory *factory;
    HPIPE hPipe;
    APIRET rc;
};

35    //*****
    //
    // MessagePipe inline members.
40    //

    inline FLAG MessagePipe::fSendMessage( TStream &stream )
    {
        return fSendMessage( stream.buffer, stream.iptr -
45    stream.buffer );
    }

    inline FLAG MessagePipe::fGetMessage( TStream &stream )
    {
50        ULONG get_len = stream.buf_len;
        if (fGetMessage( stream.buffer, &get_len )) {
            stream.iptr = stream.buffer + get_len;
            return TRUE;
        }
55        else return FALSE;
    }
```

SUBSTITUTE SHEET

```

}

inline FLAG MessagePipe::fTransact( TStream &out_strm,
5  TStream &in_strm )
{
    ULONG get_len = in_strm.buf_len;
    if (fTransact( out_strm.buffer, out_strm.iptr -
out_strm.buffer, in_strm.buffer, &get_len )) {
10     in_strm.iptr = in_strm.buffer + get_len;
        return TRUE;
    }
    else return FALSE;
}

15 #endif // MESSAGEPIPE_HPP
#ifndef OBJECTS_HPP
#define OBJECTS_HPP

#include <iomanip.h>
20 #include <TObject.HPP>

//!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
//!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
25 //
// TFlag (used for boolean fields such as StolenFlag and
InsuredFlag).
//
//     TFlag - sets initial value.
30 //
//     typecast operators:
//         FLAG - throws exception if NULL.
//         STRING - throws exception if NULL.
//
35 //     assignment operators:
//         =
//
//     comparison operators:
//         == - boolean compare. throws exception if NULL.
40 //         !=
//
//     iostream operators (friend operators).
//         <<
//         >>
45 //
//!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
//!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

#define NULL_TOK          "NULL"
50 #define TRUE_TOK         "Y"
#define FALSE_TOK         "N"

class TFlag : public virtual TNull {
55
```

SUBSTITUTE SHEET

```
public:
    TFlag();
    TFlag( FLAG flag );
    ~TFlag();

    virtual void SetDefault();

    TFlag& Assign( const TFlag& );
    TFlag& Assign( FLAG );

    operator FLAG() const;
    operator FLAG() { return value; }
    operator STRING() const;

    TFlag& operator = ( FLAG );
    TFlag& operator = ( const TFlag& );

    FLAG operator == ( const TFlag& ) const;
    FLAG operator == (      FLAG      ) const;
    FLAG operator == (      int      ) const;
    FLAG operator != ( const TFlag& ) const;
    FLAG operator != (      FLAG      ) const;
    FLAG operator != (      int      ) const;

    friend ostream& operator << ( ostream&, const TFlag&
    );
    friend istream& operator >> ( istream&,      TFlag&
    );

    friend TStream& operator << ( TStream&, const TFlag&
    );
    friend TStream& operator >> ( TStream&,      TFlag&
    );

    //----- PRIVATE IMPLEMENTATION -----
    protected:
        FLAG value;

    //fffffffffffffffffffffffffffffffffffffffffffffffffffff
    //
    // TTimestamp
    //
    // fValidate - returns TRUE if object contains a valid
    // timestamp.
    // ForceValidate - sets value to a known valid value.
    // ToSTRING - converts timestamp to a string
    // representation: "YYYY-MM-DD-HH.mm.ss.uuuuuu".
    // static fIsValidTSString - checks a string to verify
    // it's a valid timestamp.
    // UINT TSStringLen - value equals the length of the
    string representation of a timestamp.
```

SUBSTITUTE SHEET


```
//
//      manipulator operators:
//      =
5 //      +=
//
//      typecase operators:
//      STRING
//
10 //      comparison operators:
//      ==
//      !=
//      <
//      >
//      <=
15 //      >=
//
//!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!
20 struct TTimestamp : public virtual TNull {
    TTimestamp();
    TTimestamp( USHORT yr,
                UCHAR mo,
25         UCHAR dy,
                UCHAR hr = 0,
                UCHAR mn = 0,
                UCHAR sc = 0,
                USHORT ms = 0 );
    -TTimestamp();
30
    FLAG fValidate() const;
    void ForceValidate();
    STRING ToSTRING( char * ) const;
    virtual void SetDefault();
35
    static FLAG fIsValidTSString( STRING );
    static const UINT TSStringLen;

    TTimestamp& Assign( const TTimestamp& );
40     TTimestamp& Assign( USHORT, UCHAR, UCHAR, UCHAR = 0,
        UCHAR = 0, UCHAR = 0, USHORT = 0 );
    TTimestamp& Assign( STRING, FLAG isnull = FALSE );
    #ifdef __OS2__
    TTimestamp& Assign( const DATETIME& );
45 #endif

    // *** manipulator operators
    TTimestamp& operator = ( const TTimestamp& );
    #ifdef __OS2__
50     TTimestamp& operator = ( const DATETIME& );
    #endif

    operator += ( const TTimestamp& );
55
    // *** typecast opertors
```

SUBSTITUTE SHEET

```
operator STRING() const;

// *** accessors
5   USHORT usYear()      const;
   USHORT usMonth()     const;
   USHORT usDay()       const;
   USHORT usHour()      const;
   USHORT usMinute()    const;
10  USHORT usSecond()    const;
   USHORT usMillisec()  const;

// *** comparison operators
   FLAG operator == ( const TTimestamp &ts ) const;
15  FLAG operator != ( const TTimestamp &ts ) const;
   FLAG operator <  ( const TTimestamp &ts ) const;
   FLAG operator >  ( const TTimestamp &ts ) const;
   FLAG operator <= ( const TTimestamp &ts ) const;
   FLAG operator >= ( const TTimestamp &ts ) const;

20  FLAG operator == ( STRING ) const;

   friend ostream& operator << ( ostream&, const
   TTimestamp& );

25  friend TStream& operator << ( TStream&, const
   TTimestamp& );
   friend TStream& operator >> ( TStream&,
   TTimestamp& );

30  TTimestamp& AddToDate( UINT yr, UINT mon, UINT day,
   UINT hr = 0, UINT min = 0, UINT sec =
   0, UINT ms = 0 );

// Class properties.
35  static FLAG fIsLeapYear( USHORT year );
   static USHORT usMaxMonth();
   static USHORT usMaxDay( USHORT year, USHORT month );
   static USHORT usMaxHour();
40  static USHORT usMaxMinute();
   static USHORT usMaxSecond();
   static USHORT usMaxMillisec();

//----- PROTECTED IMPLEMENTATION -----
-----
45  protected:
   USHORT Year;
   UCHAR Month;
   UCHAR Day;
   UCHAR Hour;
50  UCHAR Minute;
   UCHAR Second;
   USHORT Millisec;
};
```

SUBSTITUTE SHEET

```
//ffffffffffffffffffffffffffffffffffffffffffffffff
ffffffffffffffffffffffff
//
// TString
5 //
//ffffffffffffffffffffffffffffffffffffffffffffffff
ffffffffffffffffffffffff
class TString {
10 };

#include <Objects.INL>

#endif // OBJECTS_HPP
15 #ifndef POINTER_HPP
#define POINTER_HPP

template <class T> class TPointer {

20     TPointer();

    FLAG operator !() const { return fIsNull(); }

        operator const T& () const { return
25 useAsRValue(); }
    operator      T& ()      { return
useAsLValue(); }
    const T& operator ()      () const { return
useAsRValue(); }
    T& operator ()      ()      { return
30 useAsLValue(); }
    const T* operator ->      () const { return
&useAsRValue(); }
    T* operator ->      ()      { return
&useAsLValue(); }
35     const T& operator *      () const { return
useAsRValue(); }
    T& operator *      ()      { return
useAsLValue(); }

40 // operator = () {

private:
    T *ptr;
};
45

#endif POINTER_HPP
#ifndef BITFLAGS_HPP
#define BITFLAGS_HPP

50 #include <TStream.HPP>

template <class Enum> class TBitflag {

public:
55
```

SUBSTITUTE SHEET

```
TBitflag( Enum );
TBitflag();

5 Enum Assign( Enum );

Enum Set( Enum );
Enum Clear( Enum );
Enum Change( Enum mask, Enum setting );

10 FLAG fIsSet( Enum ) const;
FLAG fIsClear( Enum ) const;
FLAG fIsAnySet( Enum ) const;
FLAG fIsAnyClear( Enum ) const;

15 Enum operator = ( Enum );

operator ULONG () const;
operator Enum () const;

20 friend TStream& operator << ( TStream&, const
TBitflag<Enum>& );
friend TStream& operator >> ( TStream&,
TBitflag<Enum>& );

25 private:
ULONG flags;
};

30 template <class Enum> TBitflag<Enum>::TBitflag( Enum e )
{
: flags( e )
}

template <class Enum> TBitflag<Enum>::TBitflag()
{
35 #ifdef DEBUG
flags = UNINIT_DATA;
#endif
}

40 template <class Enum> inline Enum TBitflag<Enum>::Assign(
Enum e )
{
return Enum( flags = e );
}

45 template <class Enum> inline Enum TBitflag<Enum>::Set(
Enum e )
{
return Enum( flags |= e );
}

50 template <class Enum> inline Enum TBitflag<Enum>::Clear(
Enum e )
{
55 return Enum( flags &= ~((ULONG)e) );
}
```

SUBSTITUTE SHEET

```

    }

    template <class Enum> inline Enum TBitflag<Enum>::Change(
    Enum mask, Enum settings )
5      {
        return Enum( flags = (flags & ~mask) | (settings &
        mask) );
    }

10     template <class Enum> inline FLAG TBitflag<Enum>::fIsSet(
    Enum e ) const
    {
        return FLAG( (flags & e) == e );
    }

15     template <class Enum> inline FLAG
    TBitflag<Enum>::fIsClear( Enum e ) const
    {
        return FLAG( (flags & e) == 0 );
20     }

    template <class Enum> inline FLAG
    TBitflag<Enum>::fIsAnySet( Enum e ) const
    {
25         return !fIsClear( e );
    }

    template <class Enum> inline FLAG
    TBitflag<Enum>::fIsAnyClear( Enum e ) const
30     {
        return !fIsSet( e );
    }

    template <class Enum> inline Enum
35     TBitflag<Enum>::operator = ( Enum e )
    {
        return Assign( e );
    }

40     template <class Enum> inline TBitflag<Enum>::operator
    ULONG () const
    {
        return flags;
    }

45     template <class Enum> inline TBitflag<Enum>::operator
    Enum () const
    {
        return (Enum)flags;
50     }

    template <class Enum> inline TStream& operator << (
    TStream &str, const TBitflag<Enum> &bf )
    {
55         return str << bf.flags;
    }
```

SUBSTITUTE SHEET

```

    }

    template <class Enum> inline TStream& operator >> (
5      TStream &str, TBitflag<Enum> &bf )
    {
        return str >> bf.flags;
    }

    #endif // BITFLAGS_HPP
10    #ifndef TBUFFER_HPP
    #define TBUFFER_HPP

    #include <iostream.h>

15    #include <debug.h>

    #include <TBitflag.HPP>
    #include <TStream.HPP>

20    #define BUFFER_UNIT          16

    //=====
    //
25    // class TBaseBuffer - implements a simple variable
    // length memory block.
    //
    class TBaseBuffer {

30    public:

        TBaseBuffer();
        TBaseBuffer( UINT bufsize );
        ~TBaseBuffer();

35        BYTE* Buf();
        const BYTE* Buf() const;

        FLAG fRealloc( UINT new_size );

40        friend TStream& operator << ( TStream&, const
        TBaseBuffer& );
        friend TStream& operator >> ( TStream&,
        TBaseBuffer& );

45    protected:

        TBaseBuffer( const TBaseBuffer& ); // Copy
        constructor.
        static UINT alloc_limit( UINT ); // Given a number,
50    returns a valid adjustment.
        BYTE* _buf() { return buffer; }
        const BYTE* _buf() const { return buffer; }
        BYTE* _newBuf( UINT new_limit );
    
```

SUBSTITUTE SHEET

```
//----- private implementation -----
//-----
private:
    BYTE *buffer;           // Beginning of
5    buffer.
    UINT limit;             // Current
    allocated buffer size.
};

10    inline BYTE* TBaseBuffer::Buf()
    {
        return buffer;
    }

15    inline const BYTE* TBaseBuffer::Buf() const
    {
        return buffer;
    }

20
//=====
//
// class TBuffer - implements a sophisticated memory
25    block.
// includes reference counting, operators, generic
// properties, etc.
//
30    class TBuffer : private TBaseBuffer {
public:

    // Type for properties of TBuffer.
    enum PROPS {
        DEFAULT = 0,
35
        FIXED    = 0x00000001, // Lock the size of the
        buffer.
        READONLY = 0x00000002, // Block any attempt to
        modify.
40        SHARED  = 0x00000004, // Changes to this string
        are shared by all.

        USER1    = 0x01000000, // User settings for
        general use.
45        USER2    = 0x02000000,
        USER3    = 0x04000000,
        USER4    = 0x08000000,
        USER5    = 0x10000000,
        USER6    = 0x20000000,
50        USER7    = 0x40000000
        // USER8    = 0x80000000 // Too big???
        (give compiler error with CSet++ 2.1)
    };
    typedef TBitflag< PROPS > TProps;
55
```

SUBSTITUTE SHEET

```
// Construction/Destruction.
TBuffer( PROPS = DEFAULT );
TBuffer( UINT length, PROPS = DEFAULT );
TBuffer( TBuffer& ); // copy
5 constructor.
~TBuffer();

// Attribute access.
10  UINT uLength() const; // Returns the
length of the buffer.
    FLAG fResize( UINT new_size ); // Shrink or grow
to a new size, returns TRUE if successful.
    void Resize( UINT new_size ); // Throws an
exception if fails.
15  const BYTE* Buf() const; // Read-only
access to data.
    BYTE* Buf(); // Access to data,
throws exeption if READONLY or !SHARED && ref_c > 1.
    const BYTE* Buf( UINT index ) const; // Returns Buf() +
20  index. checks range.
    BYTE* Buf( UINT index ); // Returns Buf() +
index. checks range.

// Reference counting.
25  UINT uRef(); // Add a
reference.
    UINT uDeref(); // Remove a
reference.
    UINT uRefCount() const; // Return the
30  reference count.
    TBuffer& PrepareToChange(); // Makes a copy of
needed (if COPYMOD=1)
    TBuffer& Copy(); // Makes a new
35  copy of this TBuffer.

// Generic property interface.
    FLAG fQueryProperty( PROPS ) const; // Returns TRUE if
specified props are set.
    PROPS SetProperty( PROPS ); // Sets specified
40  props.
    PROPS ClearProperty( PROPS ); // Clears
specified props.

// Specific property interface.
45  FLAG fQueryReadOnly() const; // TRUE if this
buffer is read-only.
    FLAG fSetReadOnly( FLAG setting );
    FLAG fQueryFixed() const; // TRUE if this
buffer's length is fixed.
50  FLAG fSetFixed( FLAG setting );
    FLAG fQueryShared() const; // TRUE if this
buffer's value is shared.
    FLAG fSetShared( FLAG setting );
55  // String functions.
```

SUBSTITUTE SHEET


```
TBuffer& StrCopy( const TBuffer& );
TBuffer& StrCopy( STRING );
TBuffer& StrConcat( const TBuffer& );
TBuffer& StrConcat( STRING );
5 TBuffer& StrTrunc( UINT index );
TBuffer& StrGrow( UINT index );
TBuffer& StrGrow( UINT index, BYTE pad );

// stream operators.
10 friend TStream& operator << ( TStream&, const TBuffer&
);
friend TStream& operator >> ( TStream&, TBuffer&
);

15 friend ostream& operator <<( ostream &os, const
TBuffer &Buf );

//----- protected implementation -----
20 protected:
// direct buffer manipulation functions.
TBuffer& _strCopy( const TBuffer& );
TBuffer& _strCopy( STRING );
TBuffer& _strConcat( const TBuffer& );
25 TBuffer& _strConcat( STRING );
TBuffer& _strTrunc( UINT index );
TBuffer& _strGrow( UINT index ); //
Grows buffer (pads with eos).
TBuffer& _strGrow( UINT index, BYTE pad ); //
30 Grows and pads buffer.

//----- private implementation -----
private:
35 // static TBufferHeap *heap; // Manages all
TBuffers.

UINT length; // Length of
40 allocated data (actual buffer is
// guaranteed 1
byte larger for eos).
UINT ref_c; // Reference Count.
TProps p_props; // Attribute
45 properties.
};

#include <TBuffer.INL>

50 #endif // TBUFFER_HPP
#ifndef TMSG_HPP
#define TMSG_HPP

typedef ULONG MSG_ID;
55 enum MSG_TYPE // Derived event classtype.
```

SUBSTITUTE SHEET

```
{
    TSYSMSG,          // TSysMsg type.
    TOBJMSG           // TObjMsg type.
}

5
//=====
//
// TMessageHandlerObject - Abstract base class for
10 TMessage aware objects.
//   AKA - TMsgHObj.
//
class TMessageHandlerObject {
15 public:

    friend class TMessage;
    typedef FLAG (TMessageHandlerObject::*
fHandleMessage)( TMessage * );

20 protected:

    virtual FLAG handleMessage( TMessage* ) = 0;
    virtual FLAG postMessage ( TMessage* ) = 0;

25 };
typedef TMessageHandlerObject TMsgHObj;           //
Define synonym.

30 //=====
//
// TMessage - Abstract base class for all messages.
//
class TMessage {
35 public:

    enum STATE {
        PRODUCED,
        POSTED,
40        PENDING,
        EXECUTING,
        CONSUMED
    };

45 TMessage( TMsgHObj *source, MSG_ID id, PVOID data );
virtual ~TMessage();

// Message Properties.
    virtual const TMsgHObj* Source() const { return
50 source; }
    virtual const STATE      State() const { return state;
}
    virtual const MSG_ID      Id() const { return id; }
    virtual const MSG_TYPE     Type() const = 0;
```

SUBSTITUTE SHEET

```
        virtual      PVOID      Data()      { return data;
    }

    // Message Methods.
5    virtual FLAG fSend() = 0;

    protected:
        STATE state;

10    private:
        TMsgHObj *source;
        MSG_ID id;
        PVOID data;
    };

15    //
    //
    //
20    class TSysMsg : public TMessage {
    public:

        static void SetSystemHandler( TMsgHObj *syshnd ) {
            system_handler = syshnd; }

25    TSysMsg( TMsgHObj *source, MSG_ID id, PVOID data );

        virtual const MSG_TYPE Type() const { return TSYSMSG;
    }

        virtual FLAG fSend();

30    private:
        static TMsgHObj *system_handler;
    };

35    class TObjMsg : public TMessage {
    public:

        TObjMsg( TMsgHObj *source, TMsgHObj *target, MSG_ID
40    id, PVOID data );

        virtual const MSG_TYPE Type() const { return TOBJMSG;
    }

        virtual FLAG fSend();

45    private:
        TMsgHObj *target;
    };

50

    class TModem : public TModem, public
    TMessageHandlerObject {
55    public:
```

SUBSTITUTE SHEET

```
FLAG handleMessage( TMessage* );
FLAG postMessage ( TMessage* );

5   };

FLAG TModem::handleMessage( TMessage *event )
{
    if (event->Source() == &Port()) {
10      if (fResultReceived()) {

          TModemMessage event = new TModemMessage( this,
rcResultCode() );

          event->fSend( ModemHandler );
15      // --> ModemHandler.handleMessage( event );
      }
    }
    else return FALSE;
20 }

FLAG TEConnect::handleModemMessage( TModemMessage *event )
{
    if (event->ResultCode() == TModem::CONNECT) {
25      waitForEnq();
      return TRUE;
    }
}

30 #endif // TMSG_HPP
#ifdef EXCEPTION_HPP
#define EXCEPTION_HPP

35 #include <iostream.h>

#include <usertype.h>

typedef ULONG ERROR_ID;

40 #define EXP_STRLIST_SIZE 10

class TException {
public:

45     enum SEVERITY {
        UNRECOVERABLE,
        RECOVERABLE
    };

50     TException( STRING string, ERROR_ID id = 0, SEVERITY =
UNRECOVERABLE );
    TException( const TException& );
    ~TException();

55     TException& AddString( STRING error_str );
```

SUBSTITUTE SHEET

```
// TException& AppendString( STRING error_str );
TException& SetSeverity( SEVERITY sev ) { severity =
sev; return *this; }
TException& SetErrorId( ERROR_ID id ) { error_id = id;
5 return *this; }

    virtual FLAG fIsRecoverable() const { return severity
== RECOVERABLE; }
    virtual STRING GetName() const { return "TException";
10 }

    STRING GetString( UINT i = 0 ) const { return
strlist[i]; }
    UINT uGetStringCount() const { return str_count; }
15 ERROR_ID GetErrorId() const { return error_id; }

private:
    STRING strlist[EXP_STRLIST_SIZE];
    UINT str_count;
20 ERROR_ID error_id;
    SEVERITY severity;
};

ostream& operator << ( ostream&, const TException& );
25

#endif // TException_HPP

#ifndef TMESSAGE_HPP
#define TMESSAGE_HPP
30

#include <usertype.h>

typedef ULONG MSG_ID;
enum MSG_TYPE // Derived event classtype.
35 {
    TSYSMSG, // TSysMsg type.
    TOBJMSG, // TObjMsg type.
    TSPECMSG // TSpecMsg type.
};
40

//=====
//
// TMessageHandlerObject - Abstract base class for
45 TMessage aware objects.
// AKA - TMsgHObj.
//
class TMessageHandlerObject {
public:
50
    friend class TMessage;
    typedef FLAG (TMessageHandlerObject::*HANDLER)(
TMessage* );
```

SUBSTITUTE SHEET

```

    FLAG fHandleMessage( TMessage* );           // Front-end
    for virtual function handlerMessage().

5   private:
    virtual FLAG handleMessage( TMessage* ) = 0; //
    Shouldn't call directly (call fHandleMessage() instead).
    };
    typedef TMessageHandlerObject TMsgHObj;      //
10   Define synonym.

    //=====
    //
15   // TMessage - Abstract base class for all messages.
    //
    class TMessage {
    public:

        enum STATE {
20             PRODUCED,           // Message has been created
            but not used.
             PENDING,             // Message has been sent and
            is pending execution.
25             EXECUTING,         // Message has been sent and
            is being executed.
             CONSUMED            // Message was consumed and
            can be destroyed.
        };

30     TMessage( TMsgHObj *source, MSG_ID id, PVOID data );
        virtual ~TMessage();

    // Message Properties.
    virtual const TMsgHObj* Source() const { return
35     source; }
        virtual const STATE    State()    const { return state;
    }
        virtual const MSG_ID    Id()       const { return id; }
        virtual const MSG_TYPE  Type()     const = 0;
40     virtual          PVOID    Data()     { return data;
    }

    // Message Methods.
    FLAG fSend();           // Front-end to the send() virtual
45     function.

    // State changes.
    void StateToPending()   { state = PENDING; }
    void StateToExecute()   { state = EXECUTING; }
50     void StateToConsumed() { state = CONSUMED; }

private:
    virtual FLAG send() = 0; // Should not call directly
    (call fSend() instead).
55
```

SUBSTITUTE SHEET

```
STATE state;
TMsgHObj *source;
MSG_ID id;
PVOID data;
5    };

    //
    //
    //
10   class TSysMsg : public TMessage {
public:

    static void SetSystemHandler( TMsgHObj *syshnd ) {
        system_handler = syshnd; }
15

    TSysMsg( TMsgHObj *source, MSG_ID id, PVOID data );

    virtual const MSG_TYPE Type() const { return TSYSMSG;
20   }

private:
    virtual FLAG send();

    static TMsgHObj *system_handler;
25   };

    //
    //
    //
30   class TObjMsg : public TMessage {
public:

    TObjMsg( TMsgHObj *source, TMsgHObj *target, MSG_ID =
35   0, PVOID data = NULL );

    virtual const MSG_TYPE Type() const { return TOBJMSG;
    }

protected:
40   virtual FLAG send();

    TMsgHObj *target;
    };

45   class TSpecMsg : public TObjMsg {
public:

    TSpecMsg( TMsgHObj *src, TMsgHObj *trt,
50   TMsgHObj::HANDLER, MSG_ID = 0, PVOID data = NULL );

    virtual const MSG_TYPE Type() const { return TSPECMSG;
    }

private:
55   virtual FLAG send();
```

SUBSTITUTE SHEET

```

    TMsgHObj::HANDLER handler;
};

/*****
5  class TModem : public TModem, public
    TMessageHandlerObject {
    public:

        FLAG handleMessage( TMessage* );
10     FLAG postMessage ( TMessage* );

};

    FLAG TModem::handleMessage( TMessage *event )
15  {
        if (event->Source() == &Port()) {
            if (fResultReceived()) {

                TModemMessage event = new TModemMessage( this,
20             rcResultCode() );

                event->fSend( ModemHandler );
                // --> ModemHandler.handleMessage( event );
            }
25         }
        else return FALSE;
    }

    FLAG TEConnect::handleModemMessage( TModemMessage *event
30  )
    {
        if (event->ResultCode() == TModem::CONNECT) {
            waitForEnq();
            return TRUE;
35         }
    }
    *****/

    #endif // TMESSAGE_HPP
40  #ifndef TMODEM_HPP
    #define TMODEM_HPP

    #include <TPort.HPP>

45  //fffffffffffffffffffffffffffffffffffffffffffffffffffff
    ffffffffffffffffffffffff
    //
    // class TModem -
    //
50  //
    //
    //fffffffffffffffffffffffffffffffffffffffffffffffffffff
    ffffffffffffffffffffffff
    class TModem {
55
```

SUBSTITUTE SHEET

public:

```
enum RC {  
    OK = 0,  
    CONNECT = 1,  
    RING = 2,  
    NO_CARRIER = 3,  
    ERROR = 4,  
    CONNECT_1200 = 5,  
    NO_DIALTONE = 6,  
    BUSY = 7,  
    NO_ANSWER = 8,  
    EXTENDED_RC = 9  
};
```

```
enum EVENT {  
    EV_ANYCMD,  
    EV_OK,  
    EV_CONNECT,  
    EV_RING,  
    EV_NOCARR,  
    EV_ERROR  
};
```

```
TModem( TPort &port );
```

```
FLAG fSendCommand( STRING );
```

```
FLAG fResultReceived();
```

```
RC rcResultCode() const;
```

```
STRING strResultCode() const;
```

```
RC rcSendCommand( STRING, ULONG timeout );
```

```
STRING strSendCommand( STRING str, ULONG timeout );
```

```
STRING strGetString( ULONG timeout );
```

```
const TPort& Port() const { return port; }
```

```
TPort& Port() { return port; }
```

```
#ifndef THREADS
```

```
void ManageEvents();
```

```
// For single
```

```
threaded usage.
```

```
#endif
```

```
//----- PRIVATE IMPLEMENTATION -----  
-----
```

```
private:
```

```
TPort& port;
```

```
char last_command[80];
```

```
char last_result[80];
```

```
RC last_rc;
```

```
};
```

SUBSTITUTE SHEET

```
#endif // TMODEM HPP
#ifndef TOBJECT_HPP
#define TOBJECT_HPP

5  #include <usertype.h>
    #include <debug.h>

    #include <TStream.HPP>

10  #include <iostream.h>

    //BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
    BBBBBBBBBBBBBBBBBBBBBB
    //
15  // CTIMS Root types.
    //
    // These types are used by derivation only. They are not
    meant to be
    //      implemented.
20  //
    //      TObject -
    //      TNull -
    //
    //BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
25  BBBBBBBBBBBBBBBBBB

    //fffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff
    ffffffffffffffffffffff
    //
30  // Object root class.
    //
    //fffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff
    ffffffffffffffffffffff
    class TObject {
35  public:

        virtual ~TObject();

40  // ...
    // not implemented.
    // ...
};

45  //fffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff
    ffffffffffffffffffffff
    //
    // CTIMS Nullable Object root class.
    //
50  // Public interface:
    //
    //      TNull - sets initial value.
    //      TRUE = object is NULL.
    //      FALSE = object has a value.
55  //      fIsNull - returns TRUE if NULL.
```

SUBSTITUTE SHEET

```
//      fSetNull - sets object to NULL.
//      fSetDefault - sets object to its default value.
//      (pure virtual).
//
5 //      operator ! - returns TRUE if object is NULL.
//
//      ///////////////////////////////////////////////////
//      ///////////////////////////////////
10 class TNull : public virtual TObject {
public:
    TNull( FLAG isnull = TRUE );

15     FLAG fIsNull() const;
    virtual FLAG fSetNull();
    virtual void SetDefault() {}          // This should be
    pure virtual!!!

20     FLAG operator !() const;

    friend TStream& operator << ( TStream&, const TNull&
    );
    friend TStream& operator >> ( TStream&,          TNull&
25 );

    //----- PROTECTED IMPLEMENTATION -----
    -----
    protected:

30     virtual void setNotNull();          // called when used
    as an L-Value.
    virtual void useAsValue() const; // called when used
    as an R-Value.

35     //----- PRIVATE IMPLEMENTATION -----
    -----
    private:
        FLAG isnull;
40     };

    #include <TObject.INL>

    #endif // TOBJECT_HPP
45     #ifndef TPOINTER_HPP
    #define TPOINTER_HPP

    #include <debug.h>

50     template <class T> class TPointer {
    public:

        TPointer() : ptr( NULL ) {}
        TPointer( T *pt ) : ptr( pt ) {}

55
```

SUBSTITUTE SHEET

```

    FLAG operator !() const { return ptr == NULL; }

        operator const T* () const { return
5    useAsRValue(); }
        operator      T* ()      { return
    useAsLValue(); }
        const T* operator ->      () const { return
10    useAsRValue(); }
        T* operator ->      ()      { return
    useAsLValue(); }
        const T& operator *      () const { return
    *useAsRValue(); }
        T& operator *      ()      { return
15    *useAsLValue(); }

    TPointer& operator = ( T* pt ) { ptr = pt; return
    *this; }

    FLAG operator == ( PVOID p ) const { return (PVOID)ptr
20    == p; }
    FLAG operator != ( PVOID p ) const { return !(*this ==
    p); }

protected:
25    const T* useAsRValue() const { ASSERT( ptr != NULL );
    return ptr; }
        T* useAsLValue()      { ASSERT( ptr != NULL );
    return ptr; }

30    private:
        T *ptr;
    };

35    #endif // TPOINTER_HPP
    #ifndef TPORT_HPP
    #define TPORT_HPP

    #include <debug.h>

40    #ifdef _OS2
        #define _THREADS
        #include "CT_Buffer.HPP"
        #include "CT_Log.HPP"
    #endif

45    #ifdef _Windows
        // Windows includes here.
    #endif

50    //!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
    //!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
    //
    // TPort - implements a com port as an object.
    //
```

SUBSTITUTE SHEET

```
// fOpenPort - opens the port, initializes it to
// desired settings.
// fClosePort - closes the port.
5 // FlushInputBuffer - flushes the input buffer.
// FlushOutputBuffer - flushes the output buffer.
// fIsEmpty - returns TRUE if the Ports buffers are
// empty.
// fIsFull - returns TRUE if the Ports buffers are
10 // full.
//
// fGetChar - gets a character from the input buffer.
// fPutChar - puts a character into the output buffer.
// fReadPort - reads a block of data from the input
15 // buffer.
// fWritePort - reads a block of data to the output
// buffer.
// fDropDTR - drops DTR (signals that the computer is
// not ready).
20 // fRaisedTR - raises DTR (signals that the computer
// is ready).
//
// StartLog - start logging all incoming characters.
// StopLog - stops logging incoming characters.
25 // fDumpLog - writes the log to a file and resets the
// log.
// rcErrorCode - returns the os specific error code
// from the last operation.
//
30 //!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
//!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
class TPort {

public:
35
#ifdef OS2
enum PARITY {
    ) = 0, // No parity.
    ODD = 1, // Odd parity.
40    EVEN = 2, // Even parity.
    MARK = 3, // Mark parity (parity bit always
1).
    SPACE = 4 // Space parity (parity bit
always 0).
45 };
enum STOP_BITS {
    ONE = 0, // 1 stop bit.
    ONE_AND_HALF = 1, // 1.5 stop bits (valid with 5
data bit length only).
50    TWO = 2 // 2 stop bits (not valid with 5
bit WORD length).
};
#endif // OS2
#ifdef Windows
55 enum PARITY {
```

SUBSTITUTE SHEET

```

    NO,                // No parity.
    ODD,               // Odd parity.
    EVEN,              // Even parity.
    MARK,              // Mark parity (parity bit always
5  1).
    SPACE              // Space parity (parity bit
always 0).
    };
    enum STOP_BITS {
10    ONE,              // 1 stop bit.
    ONE_AND_HALF,      // 1.5 stop bits (valid with 5
data bit length only).
    TWO                // 2 stop bits (not valid with 5
bit WORD length).
15    };
#endif // _Windows

    struct ComSettings {
20    STRING port_name;
    UINT port_num;
    UINT bps;
    UINT data_bits;
    PARITY parity;
    STOP_BITS stop_bits;
25    };

    TPort();
    ~TPort();

30    FLAG fOpenPort( const ComSettings &settings );
    FLAG fClosePort();

    void FlushInputBuffer();
    void FlushOutputBuffer();
35

    FLAG fIsEmpty() const;
    FLAG fIsFull() const;

    FLAG fGetChar( char &ch );
40    FLAG fPutChar( char ch );

    FLAG fReadPort( PVOID, UINT & );
    FLAG fWritePort( PVOID, UINT );
    FLAG fWritePort( PCSZ sz );
45

    FLAG fDropDTR();
    FLAG fRaiseDTR();

#ifdef _THREADS
50    FLAG fStartManageThread();
    void ManagePort();                // Default manage
thread.
    void KillManageThread();

55    FLAG fStartCommandThread( TTHREAD );
```

SUBSTITUTE SHEET

```
    FLAG fStartCommandThread( TTHREAD, PVOID data );
    void KillCommandThread();
#endif

5      void StartLog();
      void StopLog();
      FLAG fDumpLog( const char *fname );

      ULONG rcErrorCode() const;
10     //----- PRIVATE IMPLEMENTATION -----
      -----
      private:

15     #ifdef __OS2__
          HFILE hPort;
          CT_Buffer buffer;
          CT_Log log;
          int manage_thread, command_thread;
20     APIRET rc;
          FLAG fManThread, fCmdThread, log_flag;
      #endif
      #ifdef _Windows

25     // Windows variables inserted here.

      #endif
    };

30     // Include inline functions.
      #ifdef __OS2__
          #include <tport.os2>
      #endif
      #ifdef _Windows
35     #include <tport.win>
      #endif

      #endif // TPORT_HPP
      #ifndef TSTREAM_HPP
40     #define TSTREAM_HPP

          #include <usertype.h>

          #define MAX_CTMSG_SIZE      512
          #define DEF_TSTREAM_SIZE   512
45

          //
          // TStream
          //
50     class TStream {

      public:

          TStream( UINT buf_size = DEF_TSTREAM_SIZE );
55     ~TStream();
```

SUBSTITUTE SHEET

```
void Reset();

TStream& operator << ( const FLAG    );
TStream& operator << ( const USHORT );
5  TStream& operator << ( const UINT  );
TStream& operator << ( const ULONG  );
TStream& operator << ( const char*  );

TStream& operator >> ( FLAG&    );
10 TStream& operator >> ( USHORT& );
TStream& operator >> ( UINT&    );
TStream& operator >> ( ULONG&   );
TStream& operator >> ( char*&   );

15 TStream& Put( const PVOID data, UINT size );
TStream& Get(      PVOID data, UINT size );

protected:
TStream& incExtractor( UINT );
20 TStream& incInserter( UINT );

private:
ULONG buf_len;
BYTE *buffer;
25 BYTE *iptr, *xptr;

friend class MessagePipe;
// KLUDGE for DBServer.C
};
30

/*****
*****
template <class T> TStream& operator << ( TStream&, const
T& );
35 template <class T> TStream& operator >> ( TStream&,
T& );

template <class T> TStream& operator << ( TStream
&stream, const T &t )
40 {
    return stream.Put( PVOID( &t ), sizeof( T ) );
}

template <class T> TStream& operator >> ( TStream
&stream, T &t )
45 {
    return stream.Get( PVOID( &t ), sizeof( T ) );
}
*****
*****/
50

#endif // TSTREAM_HPP
#ifndef TSTRING_HPP
#define TSTRING_HPP
55
```

SUBSTITUTE SHEET


```
#include <iostream.h>

#include <usertype.h>
#include <debug.h>

5  #include <TStream.HPP>
    #include <TBuffer.HPP>

10  FLAG fIsNull( STRING str );
    FLAG fNotNull( STRING str );
    FLAG fStrCmpE( STRING str1, STRING str2 );
    FLAG fStrCmpL( STRING str1, STRING str2 );
    FLAG fStrCmpG( STRING str1, STRING str2 );
15  FLAG operator == ( STRING str1, STRING str2 );
    FLAG operator != ( STRING str1, STRING str2 );
    FLAG operator < ( STRING str1, STRING str2 );
    FLAG operator <= ( STRING str1, STRING str2 );
    FLAG operator > ( STRING str1, STRING str2 );
    FLAG operator >= ( STRING str1, STRING str2 );
20  #include <StrOps.INL>

    class TString {

25  public:

        TString();                // Constructs null
        string.
        TString( const TString & );    // Copy
        constructor.
30    TString( STRING );            // Copy
        constructor.
        TString( STRING, STRING );    // Constructs a
        concatenation of two strings.
        -TString();

35    // *** Testing functions.
        FLAG fIsAlphanumeric () const;    // TRUE if entire
        string is alpha-num.
        FLAG fIsAlphabetic   () const;    // TRUE if entire
40    string is alphabetic.
        FLAG fIsUpperCase    () const;    // TRUE if entire
        string is upper case.
        FLAG fIsLowerCase    () const;    // TRUE if entire
        string is lower case.
45    FLAG fIsWhiteSpace     () const;    // TRUE if entire
        string is whitespace.
        FLAG fIsPrintable    () const;    // TRUE if entire
        string is printable.
        FLAG fIsPunctuation  () const;    // TRUE if entire
50    string is punctuation.
        FLAG fIsControl      () const;    // TRUE if entire
        string is control characters.
        FLAG fIsGraphics     () const;    // TRUE if entire
        string is alphabetic.

55
```

SUBSTITUTE SHEET

```

    FLAG fIsASCII      () const;      // TRUE if entire
    string is ASCII.

5    FLAG fIsDigits    () const;      // TRUE if entire
    string is decimal.
    FLAG fIsHexDigits  () const;      // TRUE if entire
    string is hexadecimal.
    FLAG fIsBinaryDigits () const;    // TRUE if entire
10   string is binary.

    // *** manipulator operators.
    TString& operator = ( const TString &str );
    TString operator - (          ) const;
    TString& operator += ( STRING );
15   TString& operator &= ( STRING );
    TString& operator |= ( STRING );
    TString& operator ^= ( STRING );

    friend TString operator + ( STRING str1, STRING str2
20   );
    friend TString operator & ( STRING str1, STRING str2
    );
    friend TString operator | ( STRING str1, STRING str2
25   );
    friend TString operator ^ ( STRING str1, STRING str2
    );

    // *** accessors.
    UINT uLength() const;
30   TString subString( UINT start_pos ) const;
    TString subString( UINT startPos, UINT length, char
    pad_char = ' ' ) const;

    char& operator [] ( unsigned index );
35   const char& operator [] ( unsigned index ) const;

    // *** typecase operators.
    operator STRING      () const;
    operator unsigned char* ();
40   operator char*      ();

    // *** stream operators.
    TString& operator << ( const TString& );
    TString& operator << ( char );
45   TString& operator << ( int );
    TString& operator << ( long );

    friend TStream& operator << ( TStream&, const TString&
50   );
    friend TStream& operator >> ( TStream&,      TString&
    );

    friend ostream& operator <<( ostream &os, const
55   TString &Str );
```

SUBSTITUTE SHEET

```
// *** properties.
    FLAG fQueryReadOnly() const;           // TRUE if this
string is read-only.
    FLAG fSetReadOnly( FLAG setting );
5    FLAG fQueryFixed() const;             // TRUE if this
string's length is fixed.
    FLAG fSetFixed( FLAG setting );
    FLAG fQueryShared() const;             // TRUE if this
string's value is shared.
10    FLAG fSetShared( FLAG setting );

private:

    TString( TBuffer *pBuffer );           // Create a new
15    TString based on a TBuffer.
    void prepareToChange();                 // Called before
any change to the string is made.

    TBuffer* assignBuffer( TBuffer* );     // Assigns the new
20    buffer to the old one.

    TBuffer *buffer;                       // Pointer to
allocated memory block.
};
25

template <class base> class TSTRING {

30    };

template <UINT length, char padding> class TCharArray {

35    TCharArray();                         // Constructs
padded array.
    TCharArray( STRING );                 // STRING Copy
constructor.

40    private:

};

45    #include <TString.INL>

    #endif // TSTRING_HPP

//*****
50    *****
//
// TFlag inline members.
//

55    inline void TFlag::SetDefault()
```

SUBSTITUTE SHEET

```
{
}

5 inline TFlag& TFlag::Assign( const TFlag &flag )
{
    setNotNull();
    value = flag.value;
    return (*this);
}

10 inline TFlag& TFlag::Assign( FLAG flag )
{
    setNotNull();
    value = FLAG( flag != FALSE );
15     return (*this);
}

inline TFlag::operator FLAG() const
20 {
    useAsValue();
    return FLAG( value != FALSE );
}

25 inline TFlag::operator STRING() const
{
    useAsValue();
    return (value) ? TRUE_TOK : FALSE_TOK;
}

30 inline TFlag& TFlag::operator = ( const TFlag &flag )
{
    return Assign( flag );
}

35 inline TFlag& TFlag::operator = ( FLAG flag )
{
    return Assign( flag );
}

40 // *** Comparison operators ***

inline FLAG TFlag::operator == ( const TFlag &flag )
const
45 {
    useAsValue();
    return FLAG( value == FLAG( flag ) );
}

50 inline FLAG TFlag::operator == ( FLAG flag ) const
{
    useAsValue();
    return FLAG( value == flag );
}

55 inline FLAG TFlag::operator == ( int flag ) const
```

SUBSTITUTE SHEET

```
{
    useAsValue();
    return FLAG( value == (flag != 0) );
}

5 inline FLAG TFlag::operator != ( const TFlag &flag )
const
{
10     useAsValue();
    return FLAG( (*this == flag) == 0 );
}

inline FLAG TFlag::operator != ( FLAG flag ) const
15 {
    useAsValue();
    return FLAG( (*this == flag) == 0 );
}

inline FLAG TFlag::operator != ( int flag ) const
20 {
    useAsValue();
    return FLAG( (*this == flag) == 0 );
}

25 //*****
//*****
//
// TTimestamp inline members.
//

30 inline void TTimestamp::SetDefault()
{
    ForceValidate();
}

35 inline TTimestamp& TTimestamp::operator = ( const
TTimestamp &ts )
{
40     return Assign( ts );
}

#ifdef __OS2__
inline TTimestamp& TTimestamp::operator = ( const
45 DATETIME &Date )
{
    return Assign( Date );
}
#endif // __OS2__

50 inline USHORT TTimestamp::usYear() const
{
    return Year;
}

55 inline USHORT TTimestamp::usMonth() const
```

SUBSTITUTE SHEET

```

    {
        return Month;
    }

5   inline USHORT TTimestamp::usDay() const
    {
        return Day;
    }

10  inline USHORT TTimestamp::usHour() const
    {
        return Hour;
    }

15  inline USHORT TTimestamp::usMinute() const
    {
        return Minute;
    }

20  inline USHORT TTimestamp::usSecond() const
    {
        return Second;
    }

25  inline USHORT TTimestamp::usMillisec() const
    {
        return Millisec;
    }

30  inline FLAG TTimestamp::operator < ( const TTimestamp
    &ts ) const
    {
        return FLAG( !(*this >= ts) );
    }

35  inline FLAG TTimestamp::operator <= ( const TTimestamp
    &ts ) const
    {
        return FLAG( !(*this > ts) );
    }

40  inline FLAG TTimestamp::operator != ( const TTimestamp
    &ts ) const
    {
        return FLAG( !(*this == ts) );
    }

45  // static member.
    inline FLAG TTimestamp::fIsLeapYear( USHORT year )
50  {
        if (year % 4 && !(year % 100 || !(year % 400))) return
TRUE;
        else return FALSE;
    }

55
```

SUBSTITUTE SHEET

```
// static member.
inline USHORT TTimestamp::usMaxMonth()
{
    return 12;
}

// static member.
inline USHORT TTimestamp::usMaxHour()
{
    return 23;
}

// static member.
inline USHORT TTimestamp::usMaxMinute()
{
    return 59;
}

// static member.
inline USHORT TTimestamp::usMaxSecond()
{
    return 59;
}

// static member.
inline USHORT TTimestamp::usMaxMillisec()
{
    return 999;
}

//-----
//
// Inline members of TBuffer.
//
inline UINT TBuffer::uLength() const
{
    return length;
}

inline void TBuffer::Resize( UINT new_size )
{
    if (!fResize( new_size )) ASSERT( FALSE );
}

inline const BYTE* TBuffer::Buf() const
{
    return TBaseBuffer::Buf();
}

inline BYTE* TBuffer::Buf()
{
    ASSERT( fQueryProperty( READONLY ) == FALSE );
}
```

SUBSTITUTE SHEET

```
    ASSERT( fQueryProperty( SHARED ) == TRUE );
    uRefCount() == 1 );
    return TBaseBuffer::Buf();
5   }

    inline const BYTE* TBuffer::Buf( UINT index ) const
    {
        ASSERT( index < uLength() );
        return Buf() + index;
10   }

    inline BYTE* TBuffer::Buf( UINT index )
    {
        ASSERT( index < uLength() );
        return Buf() + index;
15   }

    inline UINT TBuffer::uRef()
    {
        return ++ref_c;
20   }

    inline UINT TBuffer::uDeref()
    {
        // Decrement ref_c. If ref_c = 0 then delete this object.
        if (--ref_c) return ref_c;
        else {
            delete this;
            return 0;
30   }
    }

    inline UINT TBuffer::uRefCount() const
    {
        return ref_c;
35   }

    inline FLAG TBuffer::fQueryProperty( PROPS prop ) const
    {
        return props.fIsSet( prop );
40   }

    inline TBuffer::PROPS TBuffer::SetProperty( PROPS prop )
    {
        return props.Set( prop );
45   }

    inline TBuffer::PROPS TBuffer::ClearProperty( PROPS prop )
    {
        return props.Clear( prop );
50   }

    inline FLAG TBuffer::fQueryReadOnly() const
55   {
```

SUBSTITUTE SHEET


```
    return props.fIsSet( READONLY );
}

5  inline FLAG TBuffer::fSetReadOnly( FLAG f )
    {
        return FLAG( ((f ? props.Set( READONLY ) :
        props.Clear( READONLY )) | READONLY) == TRUE );
    }

10  inline FLAG TBuffer::fQueryFixed() const
    {
        return props.fIsSet( FIXED );
    }

15  inline FLAG TBuffer::fSetFixed( FLAG f )
    {
        return FLAG( ((f ? props.Set( FIXED ) : props.Clear(
        FIXED )) | FIXED) == TRUE );
    }

20  inline FLAG TBuffer::fQueryShared() const
    {
        return props.fIsSet( SHARED );
    }

25  inline FLAG TBuffer::fSetShared( FLAG f )
    {
        return FLAG( ((f ? props.Set( SHARED ) : props.Clear(
        SHARED )) | SHARED) == TRUE );
    }

30  // String functions.
    inline TBuffer& TBuffer::StrCopy( const TBuffer &buf )
    {
35      return PrepareToChange()._strCopy( buf );
    }

    inline TBuffer& TBuffer::StrCopy( STRING str )
    {
40      return PrepareToChange()._strCopy( str );
    }

    inline TBuffer& TBuffer::StrConcat( const TBuffer &buf )
    {
45      return PrepareToChange()._strConcat( buf );
    }

    inline TBuffer& TBuffer::StrConcat( STRING str )
    {
50      return PrepareToChange()._strConcat( str );
    }

    inline TBuffer& TBuffer::StrTrunc( UINT index )
    {
55      return PrepareToChange()._strTrunc( index );
    }
```

SUBSTITUTE SHEET

```
    }

    inline TBuffer& TBuffer::StrGrow( UINT index )
    {
5       return PrepareToChange()._strGrow( index );
    }

    inline TBuffer& TBuffer::StrGrow( UINT index, BYTE pad )
    {
10      return PrepareToChange()._strGrow( index, pad );
    }
    //*****
    //*****
    //
15    // TNull inline members.
    //

    inline FLAG TNull::fIsNull() const
    {
20      return isnull;
    }

    inline FLAG TNull::operator ! () const
    {
25      return fIsNull();
    }

    inline void TNull::setNotNull()
    {
30      isnull = FALSE;
    }

    inline void TNull::useAsValue() const
    {
35      // This function is called when a TObject is used in such
      // a way that it must
      // have a value.
      //
      // Once the exception layer is implemented this routine
40      // will throw an exception.
      //
      ASSERT( isnull == FALSE );
    }

45    inline TStream& operator << ( TStream &stream, const
    TNull &null )
    {
        return stream << FLAG( null.isnull );
    }

50    inline TStream& operator >> ( TStream &stream, TNull
    &null )
    {
        FLAG isnl;
55      stream >> isnl;
    }
```

SUBSTITUTE SHEET

```
        if (isnl) null.fSetNull();
        else null.setNotNull();

        return stream;
5    }

#include <string.h>

10    // Private members.
    inline void TString::prepareToChange()
    {
        buffer = &buffer->PrepareToChange();
    }

15    // *** typecast operators.
    inline TString::operator STRING () const
    {
        return buffer->Buf();
20    }

    inline TString::operator unsigned char* ()
    {
        return buffer->Buf();
25    }

    inline TString::operator char* ()
    {
        return (char*)buffer->Buf();
30    }

    inline TString& TString::operator += ( STRING str )
    {
        buffer = &buffer->StrConcat( str );
35    return *this;
    }

TString operator + ( STRING str1, STRING str2 )
{
40    return TString( str1, str2 );
}

    inline UINT TString::uLength() const
    {
45    return buffer->uLength();
    }

    inline char& TString::operator [] ( unsigned index )
    {
50    prepareToChange();
        return *((char*)buffer->Buf( index ));
    }

    inline const char& TString::operator [] ( unsigned index
55    ) const
```

SUBSTITUTE SHEET

```
{
    return *((const char*)buffer->Buf( index ));
}

5 // *** friend stream operators.
  inline TStream& operator << ( TStream &buf, const TString
    &str )
  {
10     return buf << *(str.buffer);
  }

  inline TStream& operator >> ( TStream &buf, TString &str
  )
  {
15     return buf >> *(str.buffer);
  }

  inline ostream& operator << ( ostream &os, const TString
    &Str )
20  {
    return os << *(Str.buffer);
  }

  inline FLAG TString::fQueryReadOnly() const
25  {
    return buffer->fQueryReadOnly();
  }

  inline FLAG TString::fSetReadOnly( FLAG setting )
30  {
    prepareToChange();
    return buffer->fSetReadOnly( setting );
  }

  inline FLAG TString::fQueryFixed() const
35  {
    return buffer->fQueryFixed();
  }

  inline FLAG TString::fSetFixed( FLAG setting )
40  {
    prepareToChange();
    return buffer->fSetFixed( setting );
  }

45  inline FLAG TString::fQueryShared() const
  {
    return buffer->fQueryShared();
  }

50  inline FLAG TString::fSetShared( FLAG setting )
  {
    prepareToChange();
    return buffer->fSetShared( setting );
55  }
```

SUBSTITUTE SHEET

```
#include <string.h>

//fffffffffffffffffffffffffffffffffffffffffffffffffffff
5 ffffffffffffffffffffffff
//
// TPort OS/2 inline functions.
//
//fffffffffffffffffffffffffffffffffffffffffffffffffffff
10 ffffffffffffffffffffffff

inline FLAG TPort::fIsEmpty() const
{
    return buffer.fIsEmpty();
15 }

inline FLAG TPort::fIsFull() const
{
    return buffer.fIsFull();
20 }

inline FLAG TPort::fPutChar( char ch )
{
    return fWritePort( &ch, sizeof( ch ) );
25 }

inline FLAG TPort::fGetChar( char &ch )
{
    return buffer.fGetChar( ch );
30 }

inline FLAG TPort::fWritePort( PCSZ sz )
{
    return fWritePort( (PVOID)sz, strlen( sz ) );
35 }

#ifdef _THREADS
inline FLAG TPort::fStartCommandThread( TTHREAD thread )
{
    return fStartCommandThread( thread, (PVOID)this );
40 }

inline void TPort::KillManageThread()
{
    fManThread = FALSE;
45 }

inline void TPort::KillCommandThread()
{
    fCmdThread = FALSE;
50 }
#endif // _THREADS

inline void TPort::StartLog()
55 {
```

SUBSTITUTE SHEET

```
    log_flag = TRUE;
}

5 inline void TPort::StopLog()
{
    log_flag = FALSE;
}

10 inline FLAG TPort::fDumpLog( const char *fname )
{
    return log.fDumpLog( fname );
}

15 inline ULONG TPort::rcErrorCode() const
{
    return rc;
}

20
/*
*****
*****
*          bpb.h          *
*          *
25 *****
***** */

30 #ifndef _BPB_INC
#define _BPB_INC

#include <standard.h>

35 #pragma pack (1)

struct BPB {
    WORD wBytesPerSector;
    BYTE cSectorsPerCluster;
    WORD wReservedSectors;
40    BYTE cFATS;
    WORD wRootDirEntries;
    WORD wSectors;
    BYTE cMediaDescriptor;
    WORD wSectorsPerFAT;
45    WORD wSectorsPerTrack;
    WORD wHeads;
    DWORD dwHiddenSectors;
    DWORD dwHugeSectors;
50 };

#pragma pack ()

#endif
```

SUBSTITUTE SHEET

```
/*
*****
***** */

5
/*
*****
***** */
10
*      cds.h      *
*
*****
***** */

15
#ifndef      _CDS_INC
#define      _CDS_INC

#include      <dpb.h>
#include      <standard.h>

20
#pragma pack (1)

struct CDS {
    struct CDS3 {
25
    CHAR cDirectory [0x43];
    WORD wFlags;
    struct DPB _far *lpDPB;
    union {
        WORD wStartingCluster;
        DWORD lpRedirBlock;
30
    };
    WORD wUserValue;
    WORD wRootCount;
    };
    BYTE cDeviceID;
35
    void _far *lpIFS;
    WORD wIFSValue;
};

#define CDS_CDROM      0x0080
40
#define CDS_SUBST      0x1000
#define CDS_JOIN      0x2000
#define CDS_VALID      0x4000
#define CDS_REMOTE      0x8000

45
#pragma pack ()

#endif

/*
50
*****
***** */
```

SUBSTITUTE SHEET

```
/*
*****
*****
5      *      dpb.h      *
      *
*****
***** */

10     #ifndef      _DPB_INC
      #define      _DPB_INC

      #include      <driver.h>
      #include      <standard.h>

15     #pragma pack (1)

      struct DPB {
          BYTE cDrive;
          BYTE cUnit;
20         WORD wBytesPerSector;
          BYTE cClusterMask;
          BYTE cClusterShift;
          WORD wFirstFATSector;
          BYTE cFATs;
25         WORD wRootDirEntries;
          WORD wFirstDataSector;
          WORD wMaxCluster;
          WORD wSectorsPerFAT;
          WORD wRootDirSector;
30         struct DRIVER_HEADER far *lpDriver;
          BYTE cMediaDescriptor;
          BYTE cAccessFlag;
          struct DPB far *lpNext;
          WORD wNextCluster;
35         WORD wFreeClusters;
      };

      #pragma pack ( )

40     #endif

/*
*****
***** */
45

/*
*****
*****
50     *      driver.h      *
      *
*****
***** */

55     #ifndef      _DRIVER_INC
```

SUBSTITUTE SHEET


```
#define      _DRIVER_INC

#include      <standard.h>

5  #pragma pack (1)

/* Device driver header */

struct DRIVER_HEADER {
10      struct DRIVER_HEADER _far *lpNext;
      WORD wAttribute;
#ifdef _BORLANDC
      WORD *pStrategy;
      WORD *pInterrupt;
15  #else
      void _based ((_segment) _self) *pStrategy;
      void _based ((_segment) _self) *pInterrupt;
#endif
      union {
20      CHAR cName [8];
      BYTE cUnitsSupported;
      };
};

25  /* Attribute values */

#define      IS_STDIN      0x0001
#define      IS_STDOUT     0x0002
#define      IS_HUGE_BLOCK 0x0002
30  #define      IS_NUL      0x0004
#define      IS_CLOCK      0x0008
#define      INT29H_OK      0x0010
#define      IOCTL_OK       0x0040
#define      IOCTL_QUERY_OK 0x0080
35  #define      OCRM_OK      0x0800
#define      OTB_OK         0x2000
#define      FAT_REQUIRED    0x2000
#define      IOCTL_OK       0x4000
#define      IS_CHAR_DEVICE 0x8000
40

/* Device driver commands */

#define      D_INIT         0x00
#define      D_MEDIA_CHECK  0x01
45  #define      D_BUILD_BPB  0x02
#define      D_IOCTL_READ   0x03
#define      D_READ         0x04
#define      D_NONDESTRUCTIVE_READ 0x05
#define      D_INPUT_STATUS 0x06
50  #define      D_INPUT_FLUSH 0x07
#define      D_WRITE        0x08
#define      D_WRITE_WITH_VERIFY 0x09
#define      D_OUTPUT_STATUS 0x0A
#define      D_OUTPUT_FLUSH  0x0B
55  #define      D_IOCTL_WRITE 0x0C
```

SUBSTITUTE SHEET

```
#define      D_OPEN_DEVICE      0x0D
#define      D_CLOSE_DEVICE     0x0E
#define      D_REMOVABLE_MEDIA  0x0F
5  #define      D_OUTPUT_UNTIL_BUSY  0x10
   #define      D_GENERIC_IOCTL    0x13
   #define      D_GET_LOGICAL_DEVICE  0x17
   #define      D_SET_LOGICAL_DEVICE  0x18
   #define      D_IOCTL_QUERY      0x19

10  #define      MAX_DRIVER_COMMAND  0x19

   /* Driver status values */

   #define      D_DONE      0x0100
15  #define      D_BUSY      0x0200
   #define      D_ERROR      0x8000

   /* Driver error values */

20  #define      D_WRITE_PROTECTED  0x00
   #define      D_BAD_UNIT      0x01
   #define      D_NOT_READY      0x02
   #define      D_BAD_COMMAND     0x03
   #define      D_BAD_CRC        0x04
25  #define      D_BAD_HEADER      0x05
   #define      D_SEEK_FAILURE    0x06
   #define      D_BAD_MEDIA      0x07
   #define      D_SECTOR_NOT_FOUND 0x08
   #define      D_NO_PAPER      0x09
30  #define      D_WRITE_ERROR     0x0A
   #define      D_READ_ERROR     0x0B
   #define      D_GENERAL_FAILURE 0x0C
   #define      D_BAD_DISK_CHANGE 0x0F

35  /* Request header structure */

   struct REQUEST_HEADER {

       /*The format of the request header's first portion is
40   common to all
   commands. */

       BYTE cHeaderLength;
       BYTE cUnit;
45  #define      BYTE cCommand;
       WORD wStatus;
       char cReserved [8];

       /*No further fields are required for commands

50
       06h (input status) 07h (input flush)
       0Ah (output status) 0Bh (output flush)
       0Dh (open device) 0Eh (close device)
       17h (get logical device) 18h (set logical device)
55
```

SUBSTITUTE SHEET

The request header format for the remaining commands can be handled by a set of overlapping structures. */

```
5      union {

      struct {

10          /*command 00h (initialise driver)  */

          BYTE cUnitsSupported;
          void _far *lpEndOfMemory;
          union {
15      CHAR _far *lpCommandLine;
          void _far *lpBPBTable;
          };
          BYTE cDrive;
          WORD wMessageFlag;
20      };

      /* Many commands are provided with a media descriptor
      byte at the
      first location in the variable portion of the request
      header -
25      hence another set of overlapping structures.  */

      struct {

          BYTE cMediaDescriptor;
30      union {
          struct {

          /*command 01h (media check)  */
35      BYTE cChangeStatus;
          CHAR _far *lpVolumeIDForCheck;
          };
          struct {
40      /*command 02h (build BPB)  */

          void _far *lpFATSector;
          void _far *lpBPB;
45      };
          struct {

          /*Commands 03h (IOCTL Read), 04h (Read), 08h (Write),
          09h (Write with verify) and 0Ch (IOCTL Write) all
50      transfer data to or from a buffer, though only some
          of these commands require all the following fields.  */

          BYTE _far *lpBuffer;
          WORD wCount;
55      WORD wStart;
```

SUBSTITUTE SHEET

```
CHAR _far *lpVolumeIDForIO;
DWORD dwHugeStart;
};
};
5
/* Command 05h (non-destructive read) simply returns a
character
waiting for input, if one is present and requires
10 only one
field in its request header. */
CHAR cCharWaiting;
15
struct {
/*Commands 13h (Generic IOCTL) and 19h (IOCTL query)
*/
20 BYTE cCategory;
BYTE cMinorCode;
WORD wGIOCTLReserved;
BYTE _far *lpData;
25 };
};
#pragma pack (1)
30
#endif
/*
*****
***** */
35
/*
*****
*****
40 * iosys.h *
*
*****
***** */
45
#ifndef _IOSYS_INC
#define _IOSYS_INC
#include <bpb.h>
#include <standard.h>
50
#pragma pack (1)
struct IOSYSDRIVETABLE {
55 struct IOSYSDRIVETABLE _far *lpNext;
BYTE cBIOSDrive;
```

SUBSTITUTE SHEET

```

    BYTE cDOSDrive;
    struct BPB DiskBPB;
    BYTE cFileSystemFlag;
    WORD wOpenCloseCount;
5    BYTE cDeviceType;
    WORD wFlags;
    WORD wCylinders;
    struct BPB DriveBPB;
    BYTE cReserved [6];
10    BYTE cLastTrack;
    union {
    DWORD dwLastTime;
    struct {
15        WORD wPartitionFlag;
        WORD wStartingCylinder;
    };
    };
    CHAR cVolumeLabel [12];
    DWORD dwSerialNumber;
20    CHAR cFileSystem [9];
    };

#pragma pack ( )

25 #endif

/*
*****
***** */
30

/*
*****
***** *
35    *      sft.h      *
    *
*****
***** */

40 #ifndef      _SFT_INC
#define      _SFT_INC

#include      <dpb.h>
#include      <driver.h>
45 #include      <standard.h>

#pragma pack (1)

/* System File Table Header */
50 struct SFT_HEADER {
    struct _SFT_HEADER _far *lpNext;
    WORD wCount;
55 };

```

SUBSTITUTE SHEET

```
/* System File Table */

struct SFT {
5      WORD wHandles;
      WORD wAccess;
      BYTE cAttribute;
      WORD wMode;
      union {
10     struct DPB _far *lpDPB;
      struct DRIVER_HEADER _far *lpDriver;
      };
      WORD wStartingCluster;
      WORD wTime;
      WORD wDate;
15     DWORD dwSize;
      DWORD dwFilePointer;
      WORD wRelativeCluster;
      DWORD dwDirSector;
      BYTE cDirSectorEntry;
20     CHAR cName [11];
      struct SFT _far *lpNextShare;
      WORD wMachine;
#ifdef _BORLANDC
25     void _seg *spOwner;
      WORD pSharingRecord;
#else
      _segment spOwner;
      void _based (void) *pSharingRecord;
#endif
30     WORD wAbsoluteCluster;
      void _far *lpIFS;
};

#pragma pack ( )
35

#endif

/*
40 *****
***** */

/*
45 *****
***** *
*   standard.h   *
*
*****
50 *****
***** */

#ifndef _STANDARD_INC
#define _STANDARD_INC

55 /* Logical operators and values */
```

SUBSTITUTE SHEET

```
#define      AND      &&
#define      NOT      !
#define      OR       ||

5  #define      FALSE  0
   #define      TRUE   1    // for consistency with TRUE =
   NOT FALSE

   #define      OFF    0
10  #define      ON     1

   #define      CLEAR  0
   #define      SET    1

15  /* Convenient data types */

   typedef unsigned charBYTE;
   typedef unsigned shortWORD;
   typedef unsigned longDWORD;

20  typedef signed charSBYTE;
   typedef signed intSWORD;
   typedef signed longSDWORD;

25  typedef unsigned charCHAR;

   typedef intBOOL;

   /* Macro for generating a far pointer from segment and
30  offset*/

   #ifndef MK_FP
       #define MK_FP(seg,off) (((_segment) (seg)) :> ((void
   _based (void) *) (off)))
35  #endif

   /* The above form for MK_FP has a problem (at least in C
   6.00) with
       multiple dereferencing through structures. On the
40  other hand, the
       compiler generates much more efficient code with it.
   As an alternative,
       keep the more familiar macro on standby. */

45  #if FALSE
   #define MK_FP(seg,off) ((void _far *) (((DWORD) (seg) <<
   16) | ((WORD) (off))))
   #endif

50  /* Macros to decompose 16-bit and 32-bit objects into
   high and low
       components and to reconstitute them */

   #define HIGHBYTE(x) ((BYTE) ((x) >> 8))
55  #define LOWBYTE(x) ((BYTE) (x))
```

SUBSTITUTE SHEET

```

5      #define MK_WORD(high,low) (((WORD) (high) << 8) | (low))

      #define HIGHWORD(x) ((WORD) ((x) >> 16))
      #define LOWWORD(x) ((WORD) (x))

10     #define MK_DWORD(high,low) (((DWORD) (high) << 16) |
      (low))

      /* Macros for directing the compiler to use current
      segment register
      values rather than generate relocatable references*/

      #define CODESEG _based (_segname ("CODE"))
      #define CONSTSEG _based (_segname ("CONST"))
15     #define DATASEG _based (_segname ("DATA"))
      #define STACKSEG _based (_segname ("STACK"))

      /* Macro for NULL in case using STDLIB.H would be
      inappropriate */
20     #ifndef NULL
      #define NULL ((void *) 0)
      #endif

25     #endif

      /*
      *****
      ***** */
30     ;

      *****
      *****
35     ; * driver.inc *
      ;

      *****
      *****

40     ; Device driver header

      DRIVER_HEADERSTRUCT
      lpNextdd0FFFFFFFFh
      wAttributedw0000h
45     pStrategydw0000h
      pInterruptdw0000h
      UNION
      cNamedb" "
      cUnitsSupporteddb?
50     ENDS
      DRIVER_HEADERENDS

      ; Attribute values

55     IS_STDINEQU00001h
```

SUBSTITUTE SHEET


```
IS_STDOUTEQU0002h
IS_HUGE_BLOCKEQU0002h
IS_NULEQU0004h
IS_CLOCKEQU0008h
5 INT29H_OKEQU0010h
GIOCTL_OKEQU0040h
GIOCTL_QUERY_OK EQU0080h
OCRM_OK EQU0800h
OTB_OKEQU2000h
10 FAT_REQUIRED EQU2000h
IOCTL_OKEQU4000h
IS_CHAR_DEVICE EQU8000h

; Device driver commands - these do not follow the
15 upper case convention
; because they are used to generate the names of the
procedures for each
; driver command.

20 D_INITEQU00h
D_MEDIA_CHECKEQU01h
D_BUILD_BPBEQU02h
D_IOCTL_READ EQU03h
D_READ EQU04h
25 D_NONDESTRUCTIVE_READ EQU05h
D_INPUT_STATUSEQU06h
D_INPUT_FLUSH EQU07h
D_WRITE EQU08h
D_WRITE WITH VERIFY EQU09h
30 D_OUTPUT_STATUS EQU0Ah
D_OUTPUT_FLUSH EQU0Bh
D_IOCTL_WRITE EQU0Ch
D_OPEN_DEVICE EQU0Dh
D_CLOSE_DEVICE EQU0Eh
35 D_REMOVABLE_MEDIA EQU0Fh
D_OUTPUT_UNTIL_BUSY EQU10h
D_GENERIC_IOCTL EQU13h
D_GET_LOGICAL_DEVICE EQU17h
D_SET_LOGICAL_DEVICE EQU18h
40 D_IOCTL_QUERY EQU19h

MAX_DRIVER_COMMAND EQU19h

; Driver status values
45 D_DONE EQU0100h
D_BUSY EQU0200h
D_ERROR EQU8000h

50 ; Driver error values

D_WRITE_PROTECTED EQU00h
D_BAD_UNIT EQU01h
D_NOT_READY EQU02h
55 D_BAD_COMMAND EQU03h
```

SUBSTITUTE SHEET

```

D_BAD_CRCEQU04h
D_BAD_HEADEREQU05h
D_SEEK_FAILUREEQ06h
D_BAD_MEDIAEQ07h
5 D_SECTOR_NOT_FOUNDEQU08h
D_NO_PAPEREQ09h
D_WRITE_ERRORREQ0Ah
D_READ_ERRORREQ0Bh
D_GENERAL_FAILUREEQ0Ch
10 D_BAD_DISK_CHANGE EQ0Fh

; Request Header structure

REQUEST_HEADERSTRUCT
15 cHeaderLength db?
cUnit db?
cCommanddb?
wStatusdw?
cReserveddb08h DUP (?)
20 UNION
STRUCT
cUnitsSupporteddb?
lpEndOfMemorydd?
UNION
25 lpCommandLinedd?
lpBPBTabledd?
ENDS
cDrivedb?
wMessageFlagdw?
30 ENDS
STRUCT
cMediaDescriptordb?
UNION
STRUCT
35 cChangeStatus db?
lpVolumeIDForCheckdd?
ENDS
STRUCT
40 lpFATSectordd?
lpBPB dd?
ENDS
STRUCT
lpBufferdd?
wCountdw?
45 wStartdw?
lpVolumeIDForIOdd?
dwHugeStartdd?
ENDS
ENDS
50 ENDS
cCharWaitingdb?
STRUCT
cCategory db?
cMinorCodedb?
55 wGIOCTLReserveddw?
```

SUBSTITUTE SHEET

```

        lpDatadd?
        ENDS
    ENDS
REQUEST_HEADERENDS
5
;
*****
*****

10
;
*****
*****
;    *    sft.inc    *
15
;
*****
*****

;    System File Table Header
20
SFT_HEADERSTRUCT
    lpNextdd0FFFFFFFFh
    wCountdw0000h
SFT_HEADERENDS
25
;    System File Table (with default initialisation
suitable for use with
;    FCBs)

30
SFTSTRUCT
    wHandlesdw0000h
    wAccessdw'AA'
    cAttributedb'A'
    wMode dw'AA'
35
    UNION
        lpDPBdd'AAAA'
        lpDriverdd'AAAA'
    ENDS
    wStartingClusterdw'AA'
40
    wTime dw'AA'
    wDate dw'AA'
    dwSizedd'AAAA'
    dwFilePointer dd00000000h
    wRelativeClusterdw'AA'
45
    dwDirSectordd'AAAA'
    cDirSectorEntrydb'A'
    cName db'AAA.AAAAAA.A'
    lpNextSharedd'AAAA'
    wMachinedw'AA'
50
    spOwnerdw'AA'
    pSharingRecorddw'AA'
    wAbsoluteClusterdw'AA'
    lpIFS dd'AAAA'
SFTENDS
55
```

SUBSTITUTE SHEET

```

;
*****
*****
5
;
*****
*****
10
; * standard.inc *
;
*****
*****

.NOCREF standard_inc
15 IFNDEFstandard_inc

; Logical symbols

.NOCREF FALSE, TRUE
20 FALSEEQUO
TRUEEQU(NOT FALSE)

standard_inc = TRUE
25 ENDIF

;
*****
*****
30
```

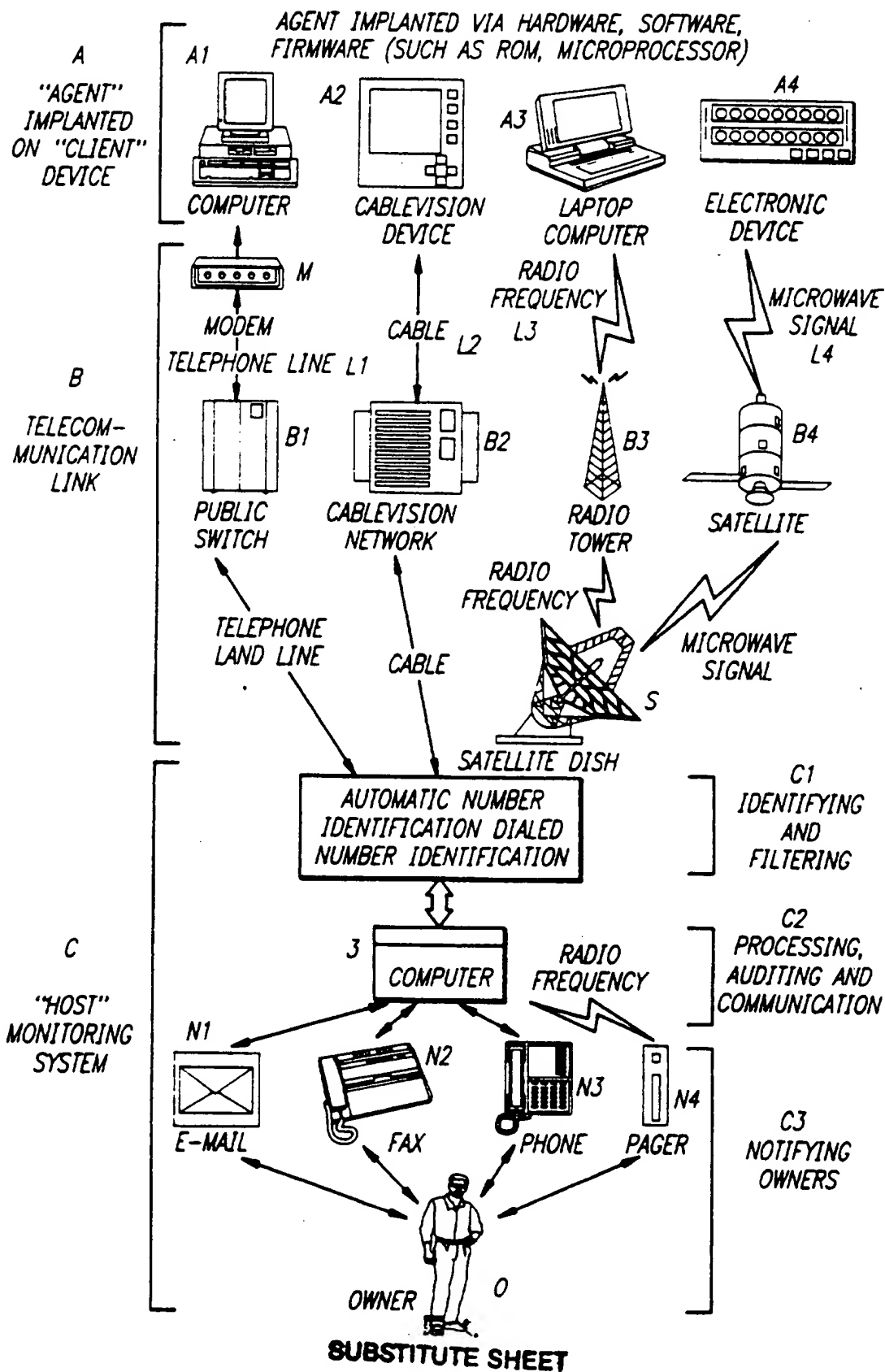
SUBSTITUTE SHEET

WHAT IS CLAIMED IS:

1. An apparatus with an integral computer tracing and security monitoring system comprising a transparent agent controlling means for sending signals to a host monitoring system via a telecommunication link, at spaced-apart intervals of time, said signals including identifying indicia for said device.
2. An apparatus as claimed in claim 1, wherein the means for sending signals includes a telecommunication interface connectable to a communication link.
3. An apparatus as claimed in claim 1, wherein the means sends signals at regular periodic intervals.
4. A computer tracing and security monitoring system, comprising:
a computer;
a telecommunication interface operatively connected to the computer; and
means controlled by the computer for sending signals to the telecommunication interface including signals for contacting a host monitoring system, and for providing the host monitoring system with identification indicia.
5. A system as claimed in claim 4, wherein the computer has addressable memory (such as read-only memory or random-access memory, and the means includes software.
6. A method for providing a computer with a Agent security system, comprising the steps of preparing software for the computer with instructions for dialing a host monitoring system number without visual or audible signals and transmitting identification indicia, and programming the software into addressable memory of the computer at a location not normally accessible to operating software for the computer.

SUBSTITUTE SHEET

FIG. 1



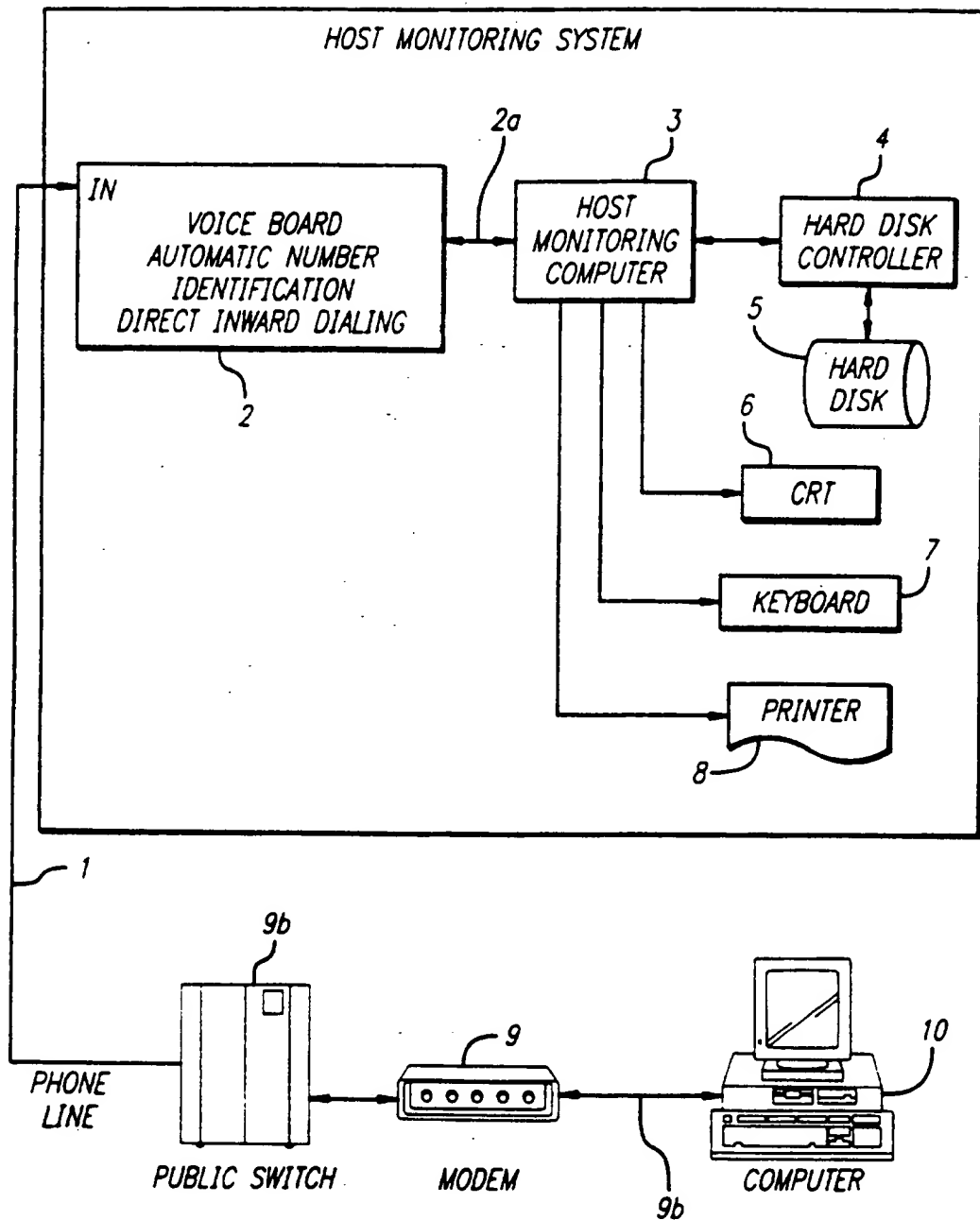


FIG. 2

SUBSTITUTE SHEET

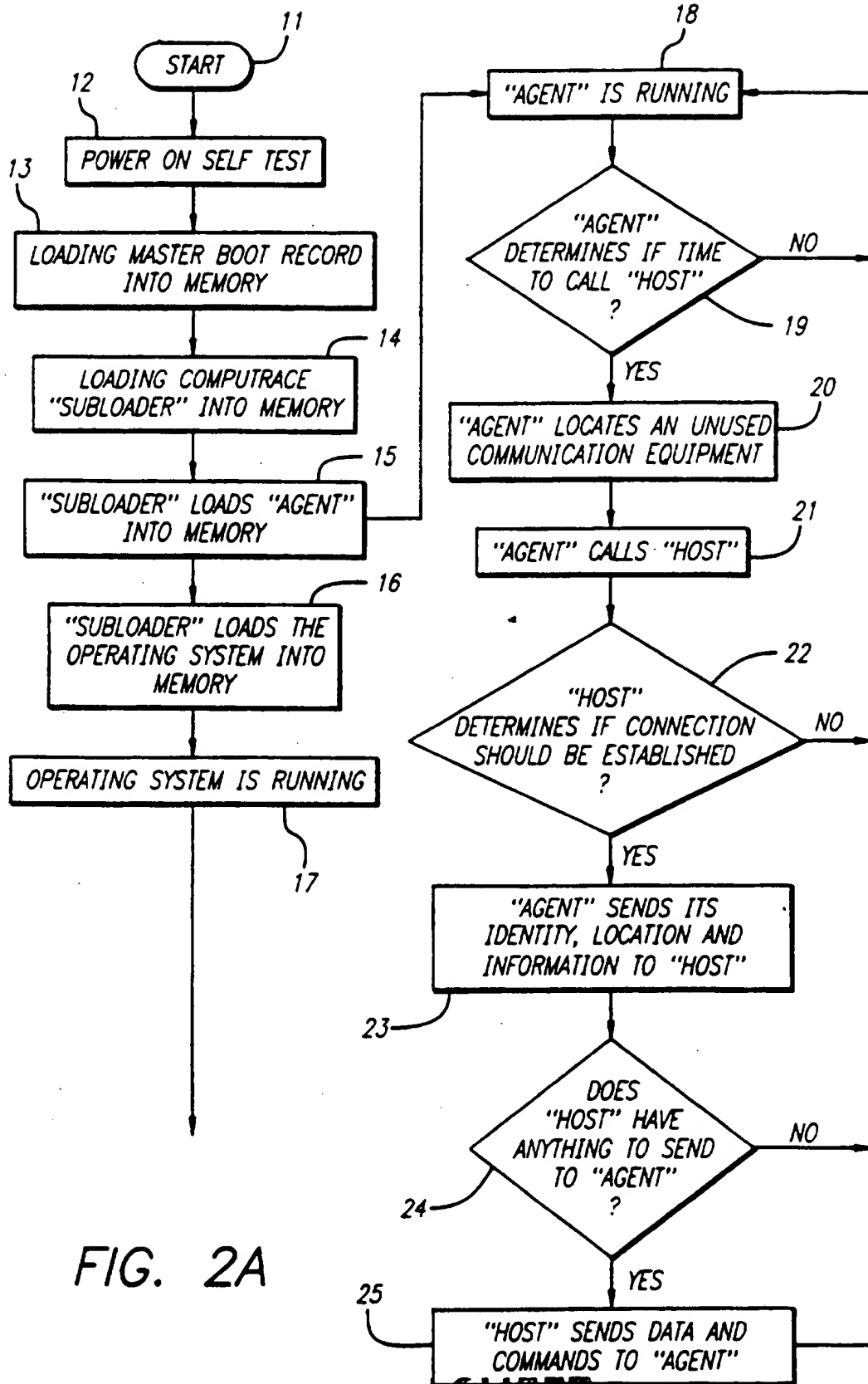
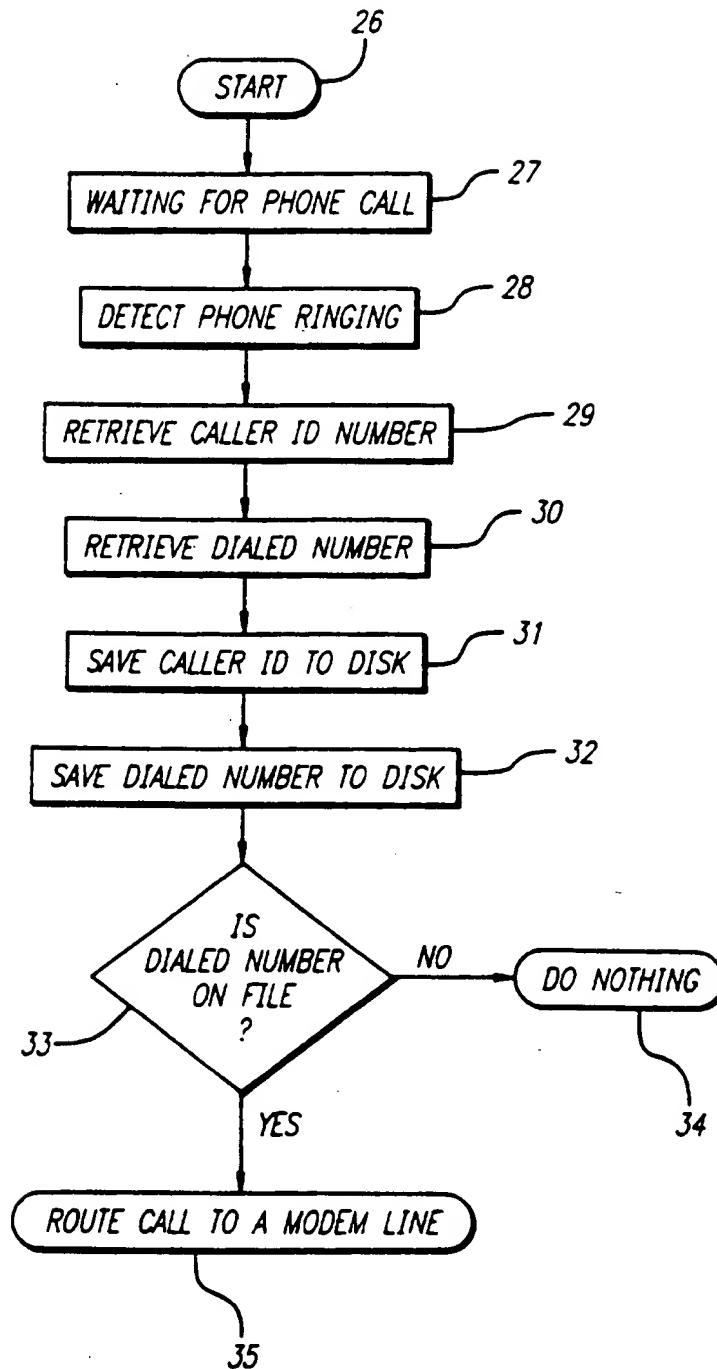


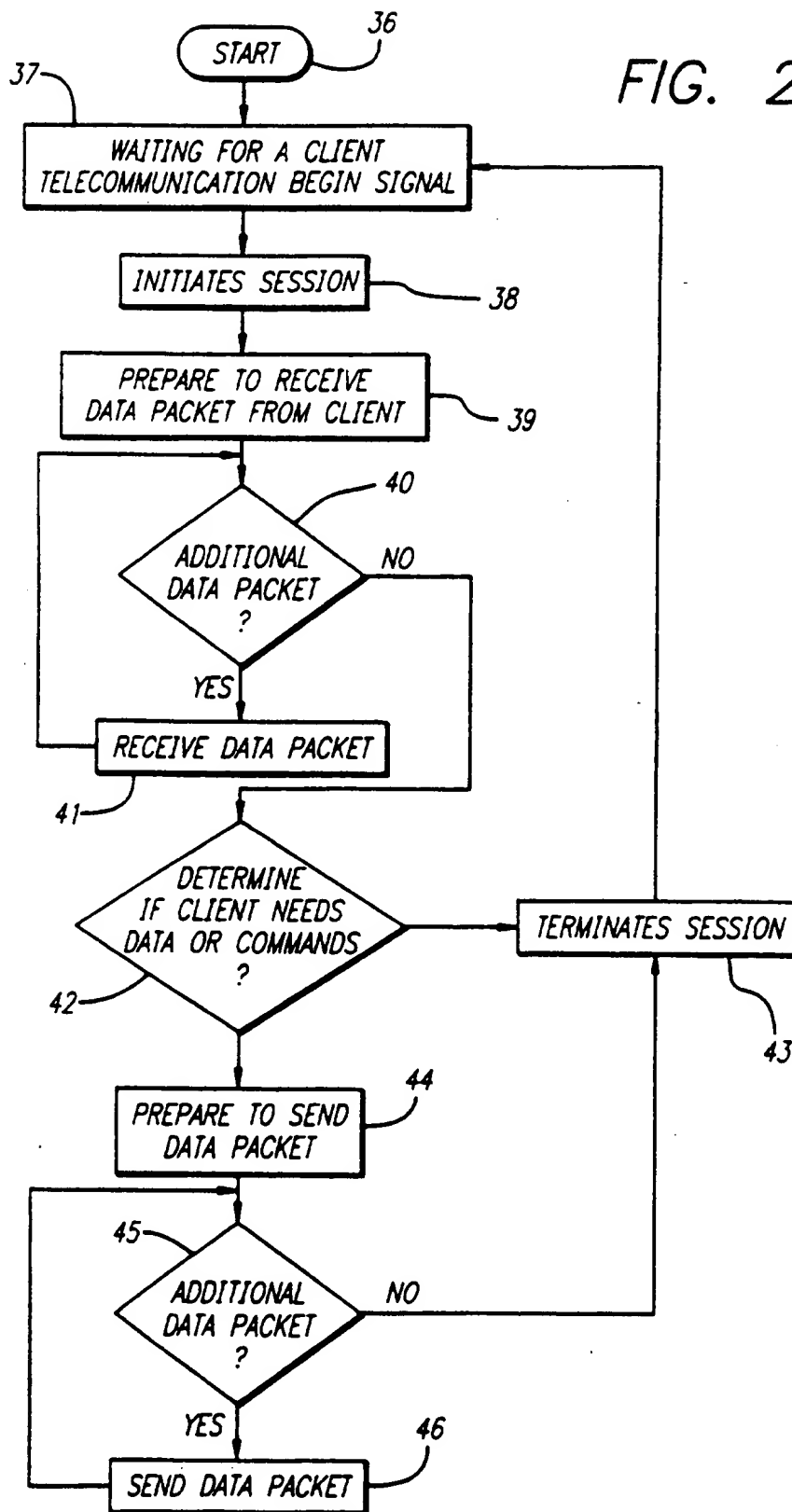
FIG. 2A

FIG. 2B

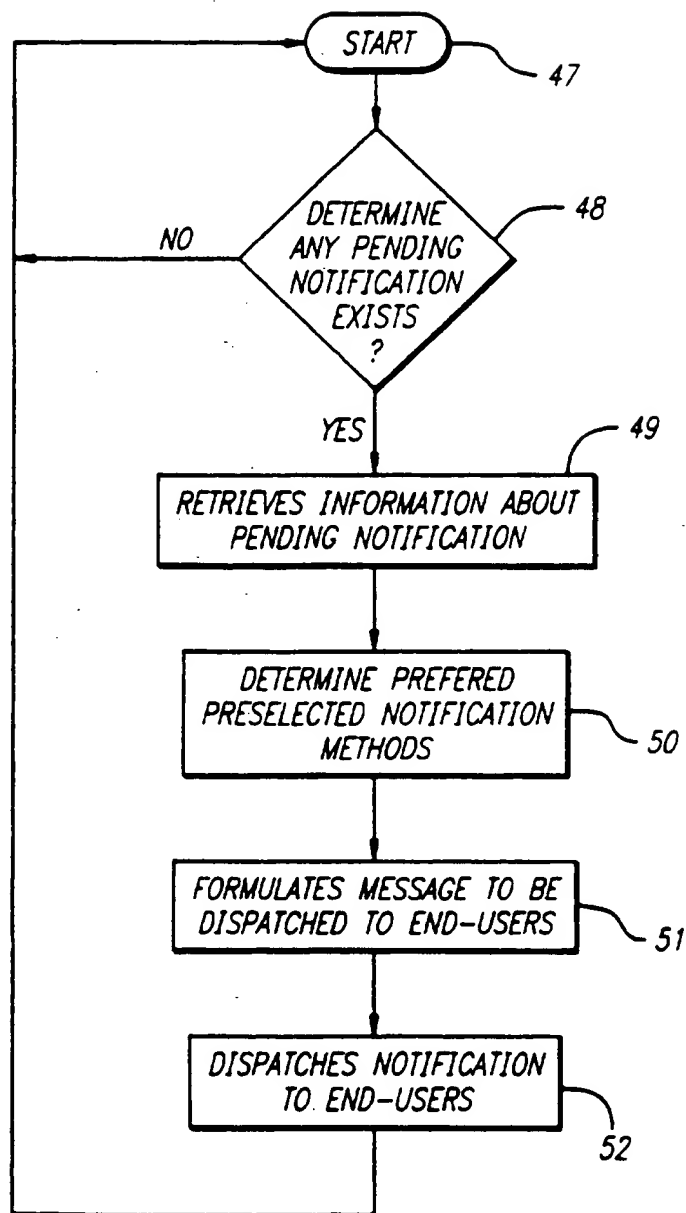


SUBSTITUTE SHEET

FIG. 2C



SUBSTITUTE SHEET

*FIG. 2D*

SUBSTITUTE SHEET

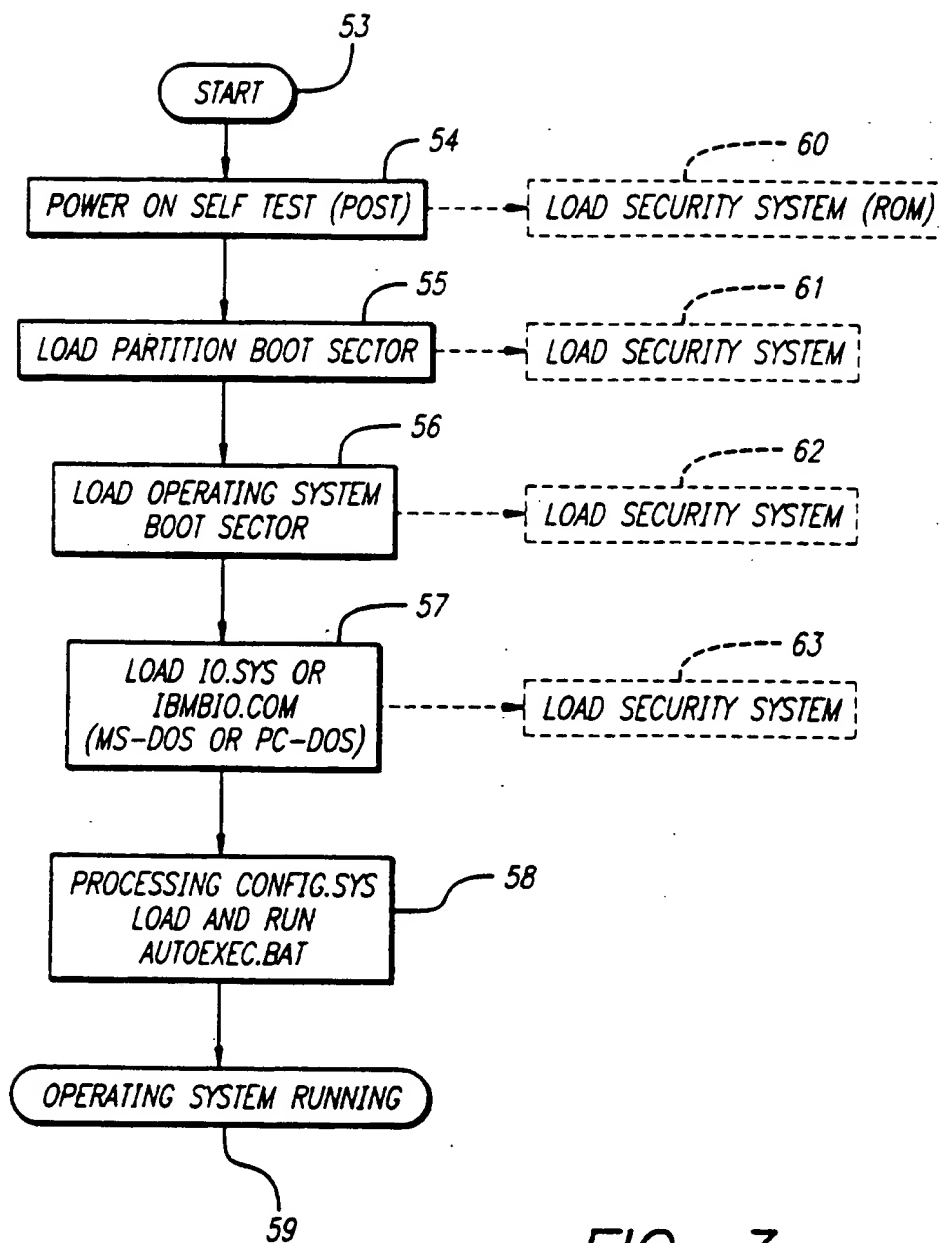
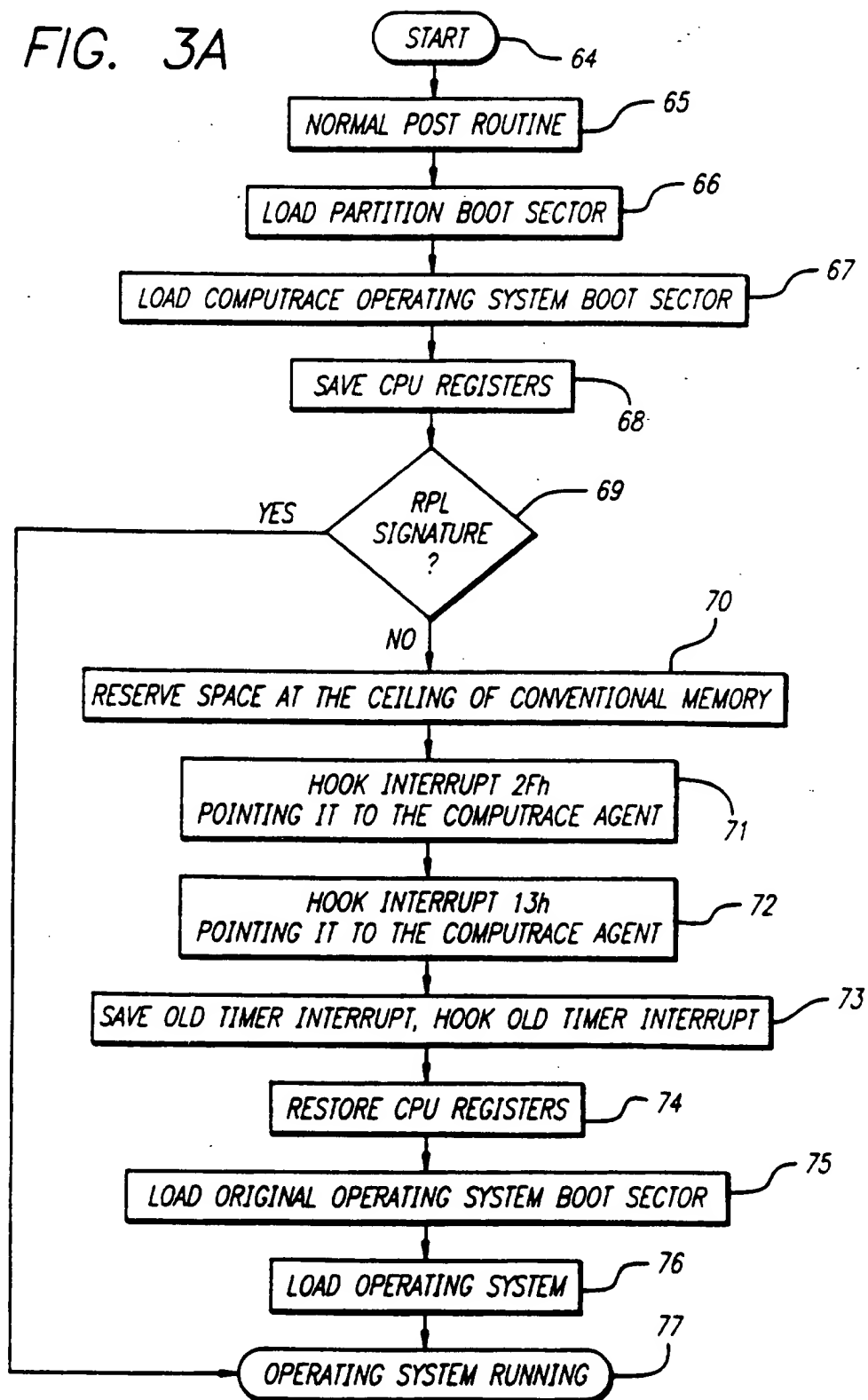


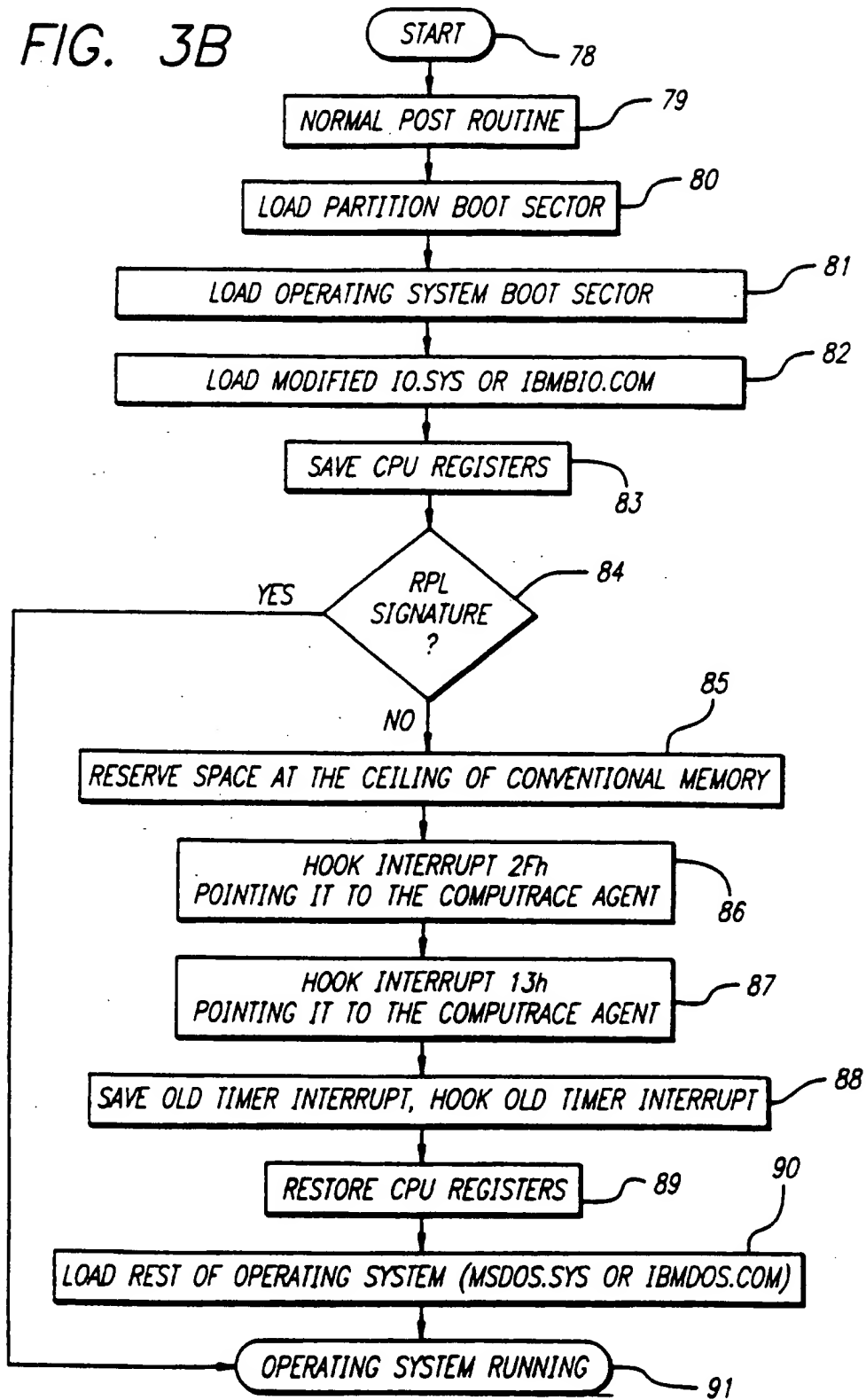
FIG. 3

FIG. 3A



SUBSTITUTE SHEET

FIG. 3B



SUBSTITUTE SHEET

FIG. 3C

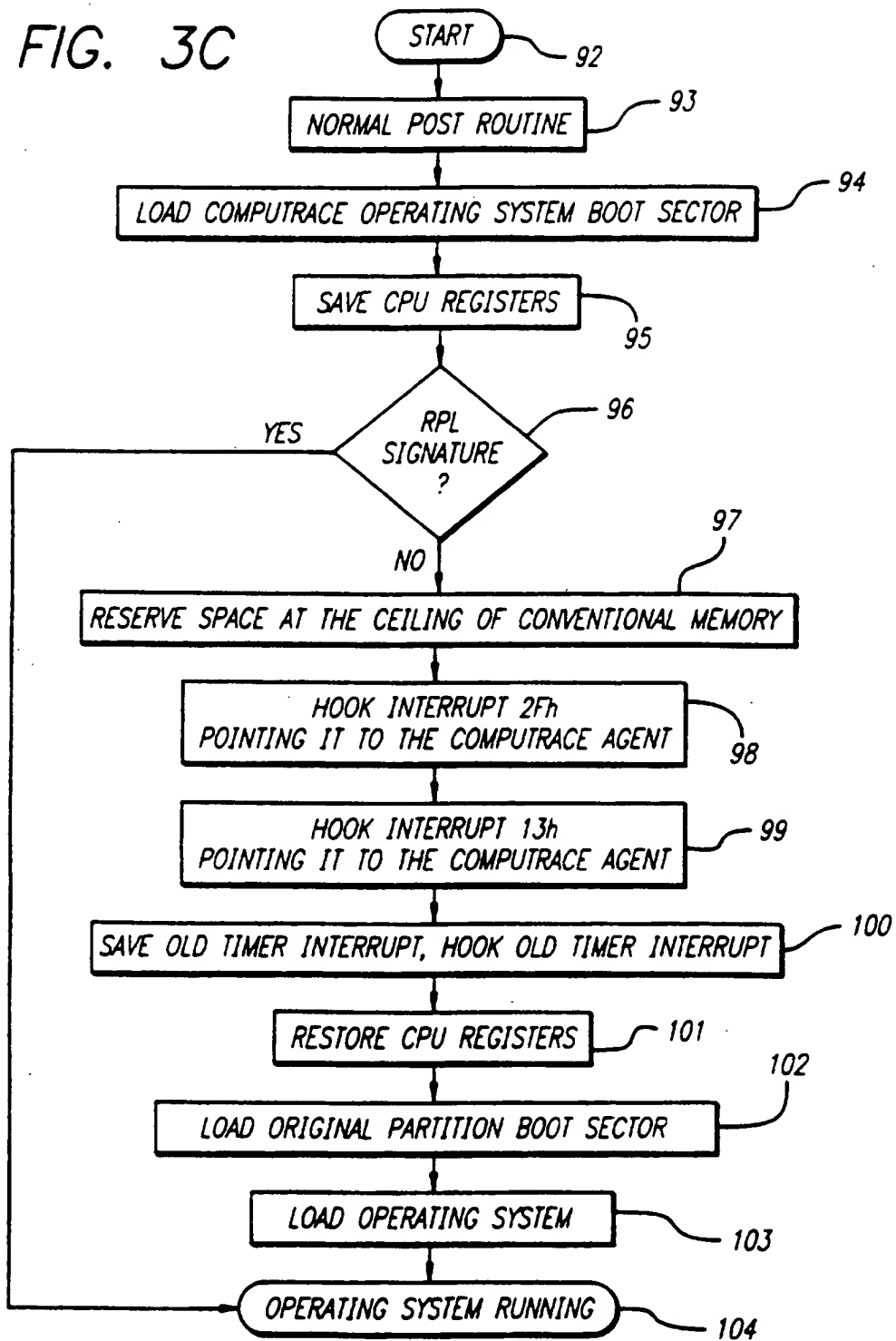
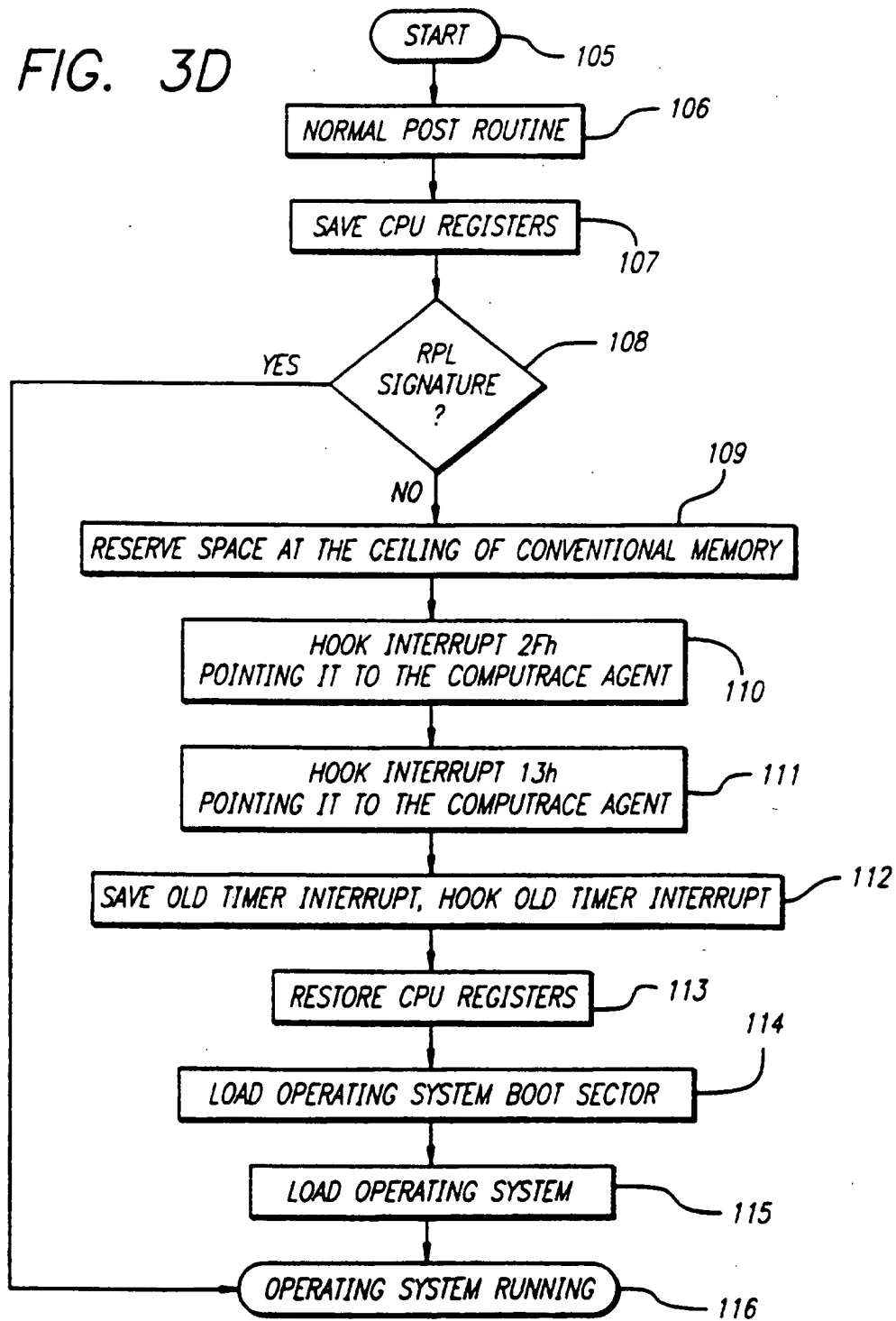
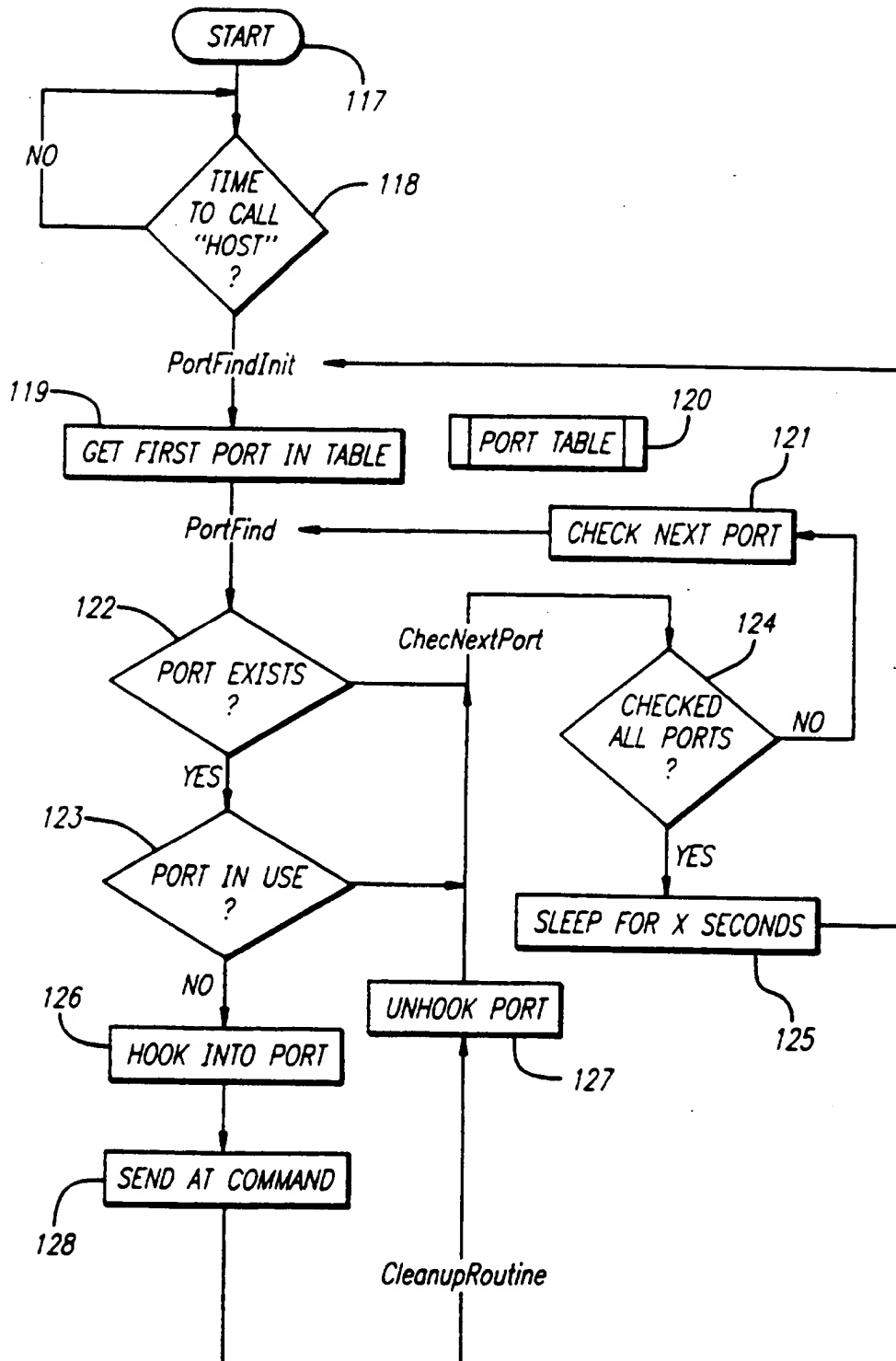


FIG. 3D



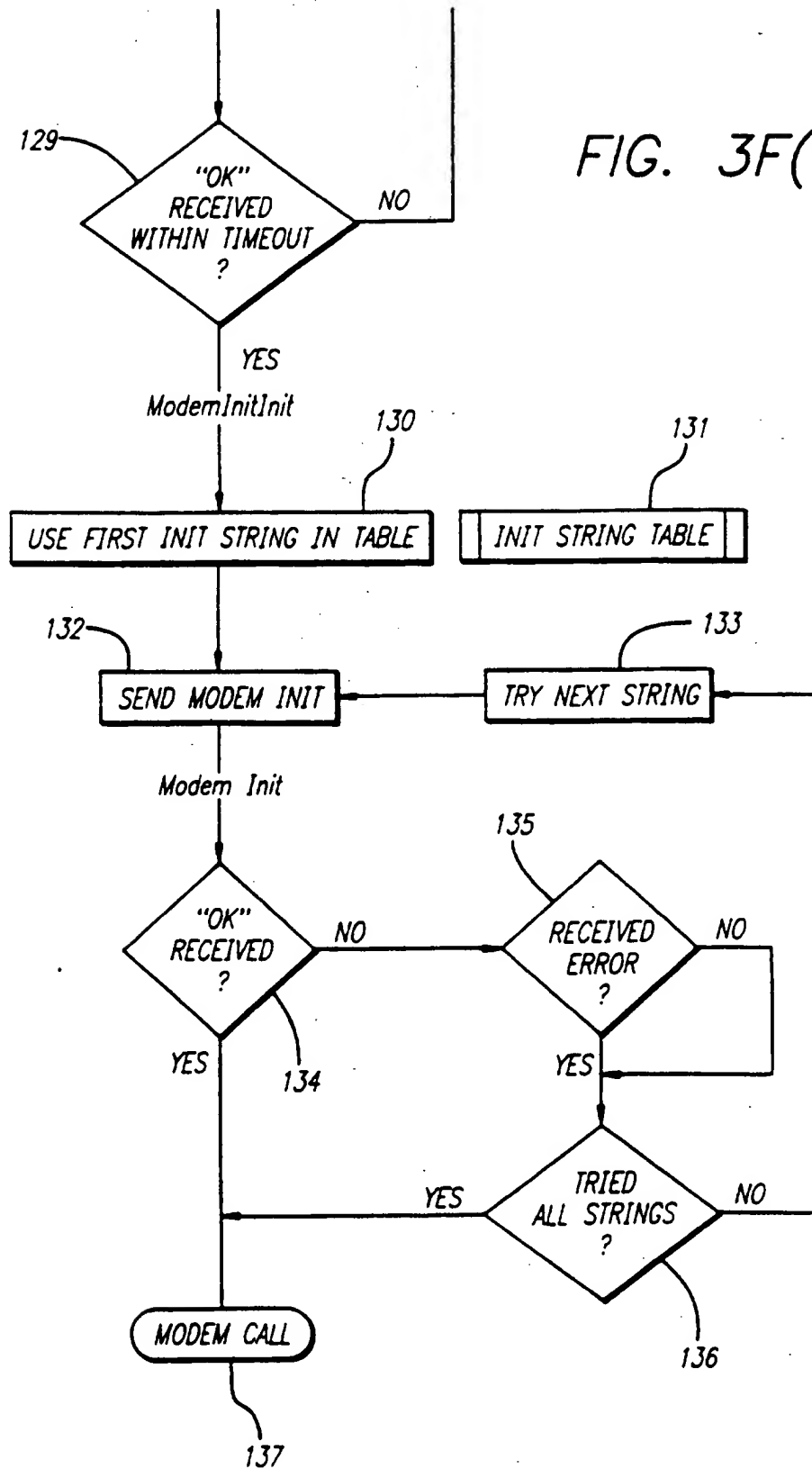
SUBSTITUTE SHEET

FIG. 3F(1)



SUBSTITUTE SHEET

FIG. 3F(2)



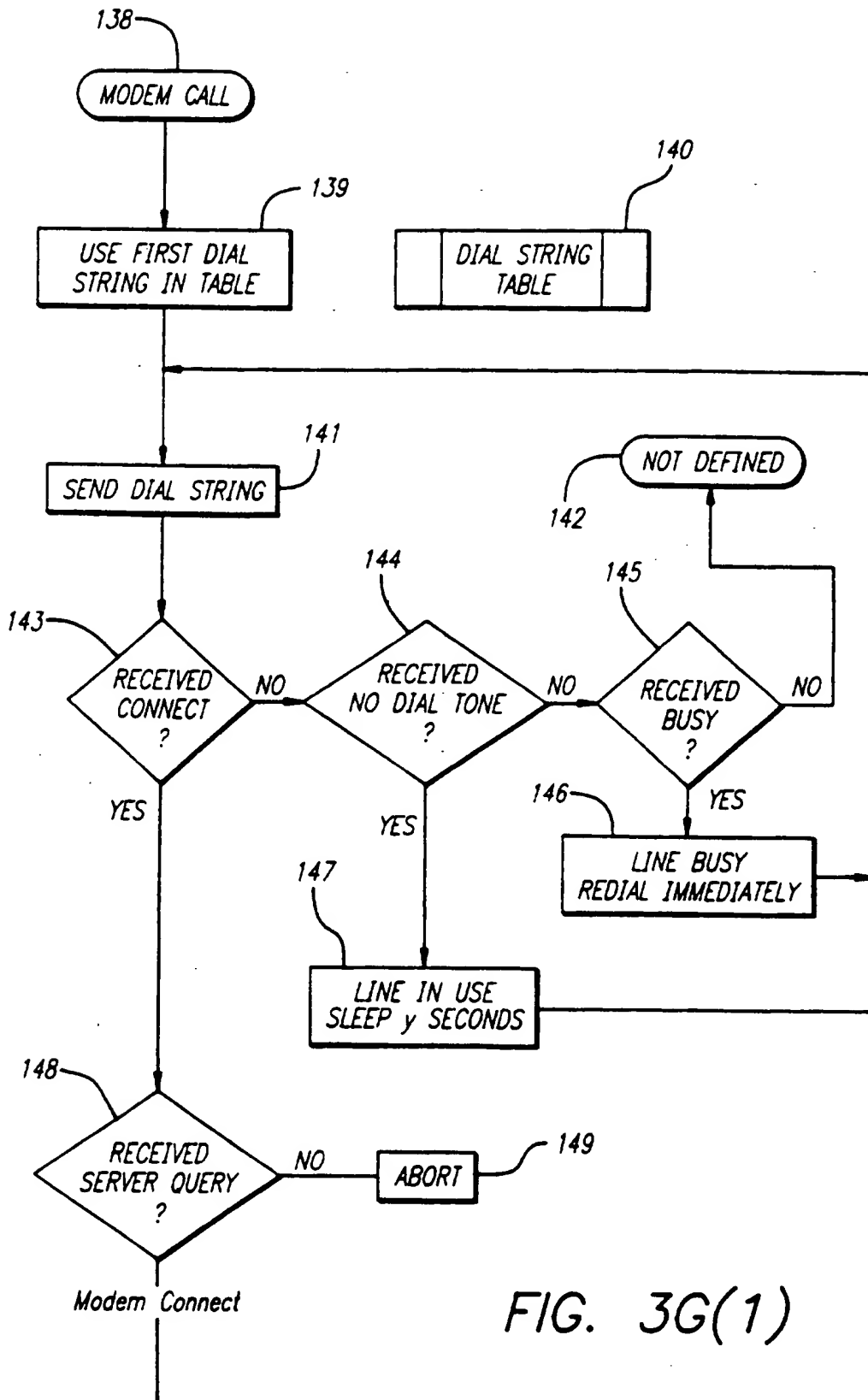


FIG. 3G(1)

FIG. 3G(2)

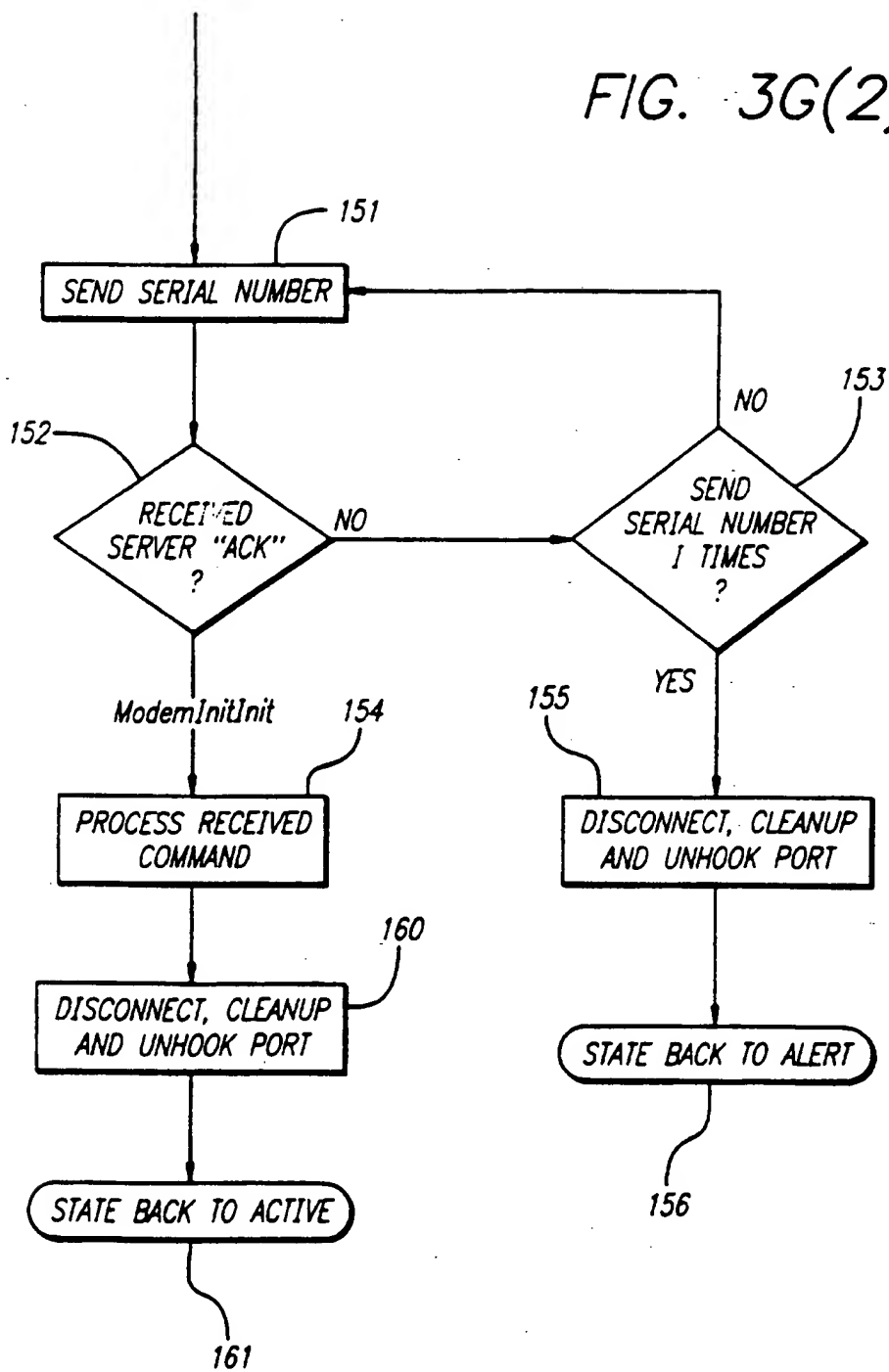


FIG. 3H

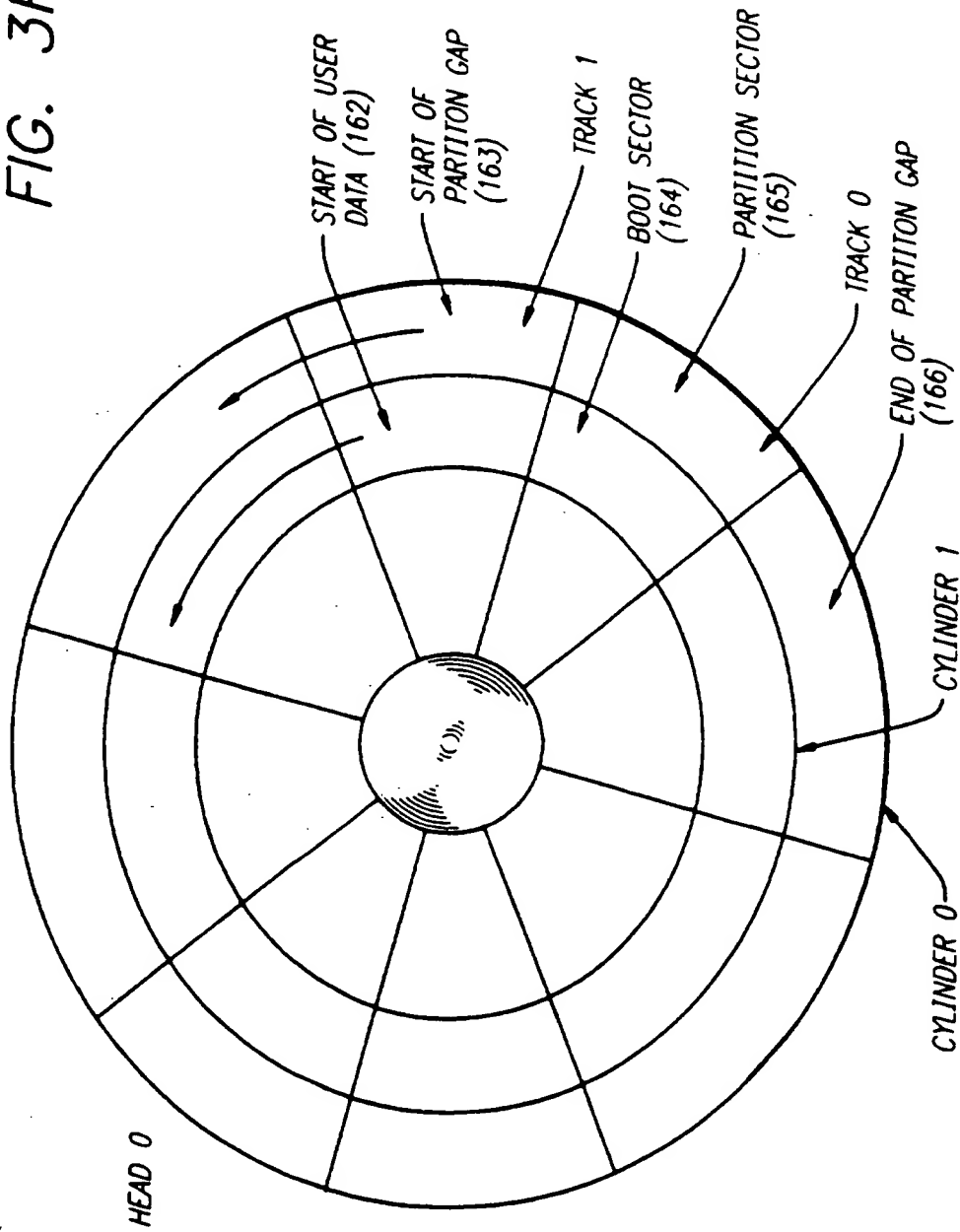
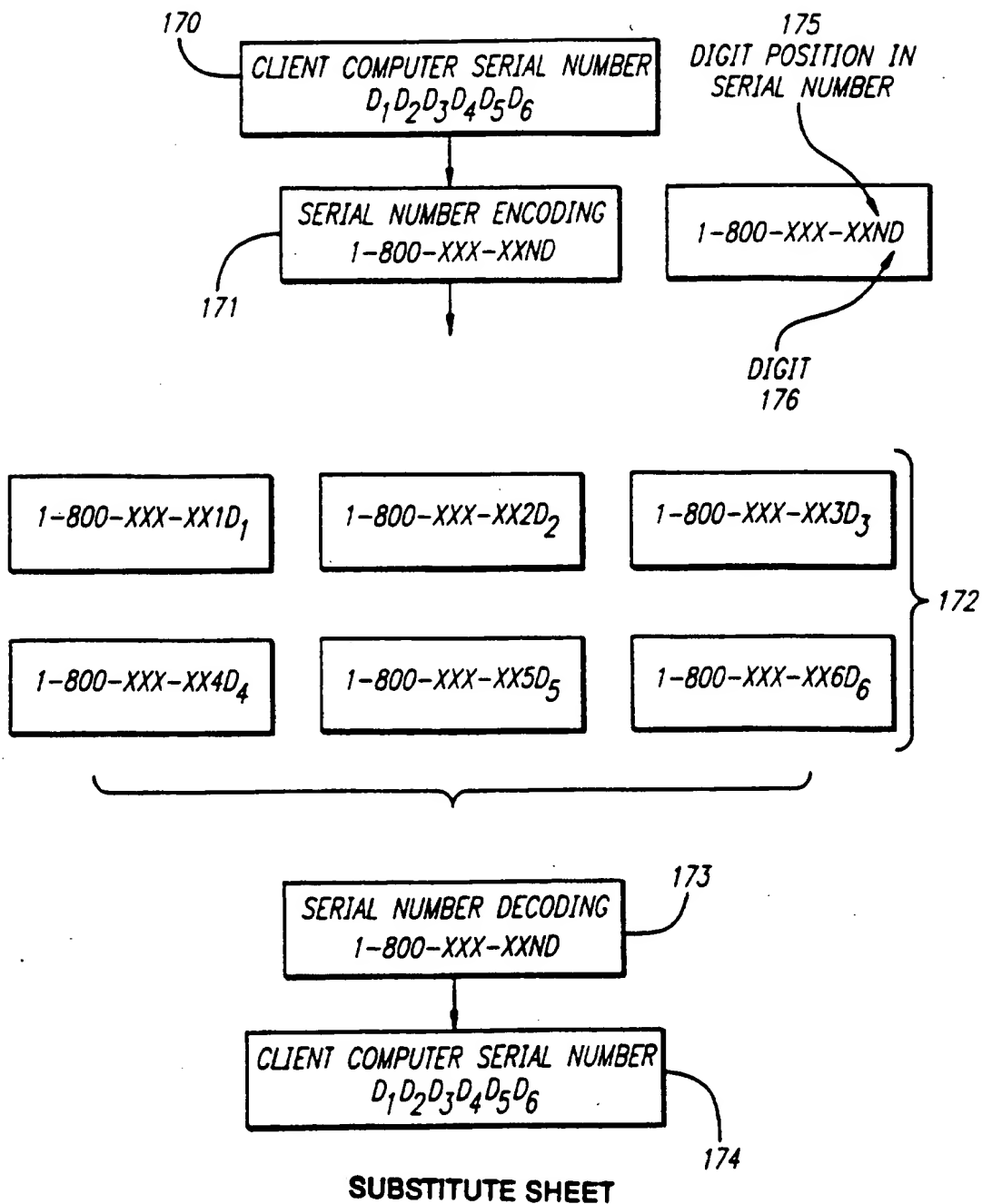
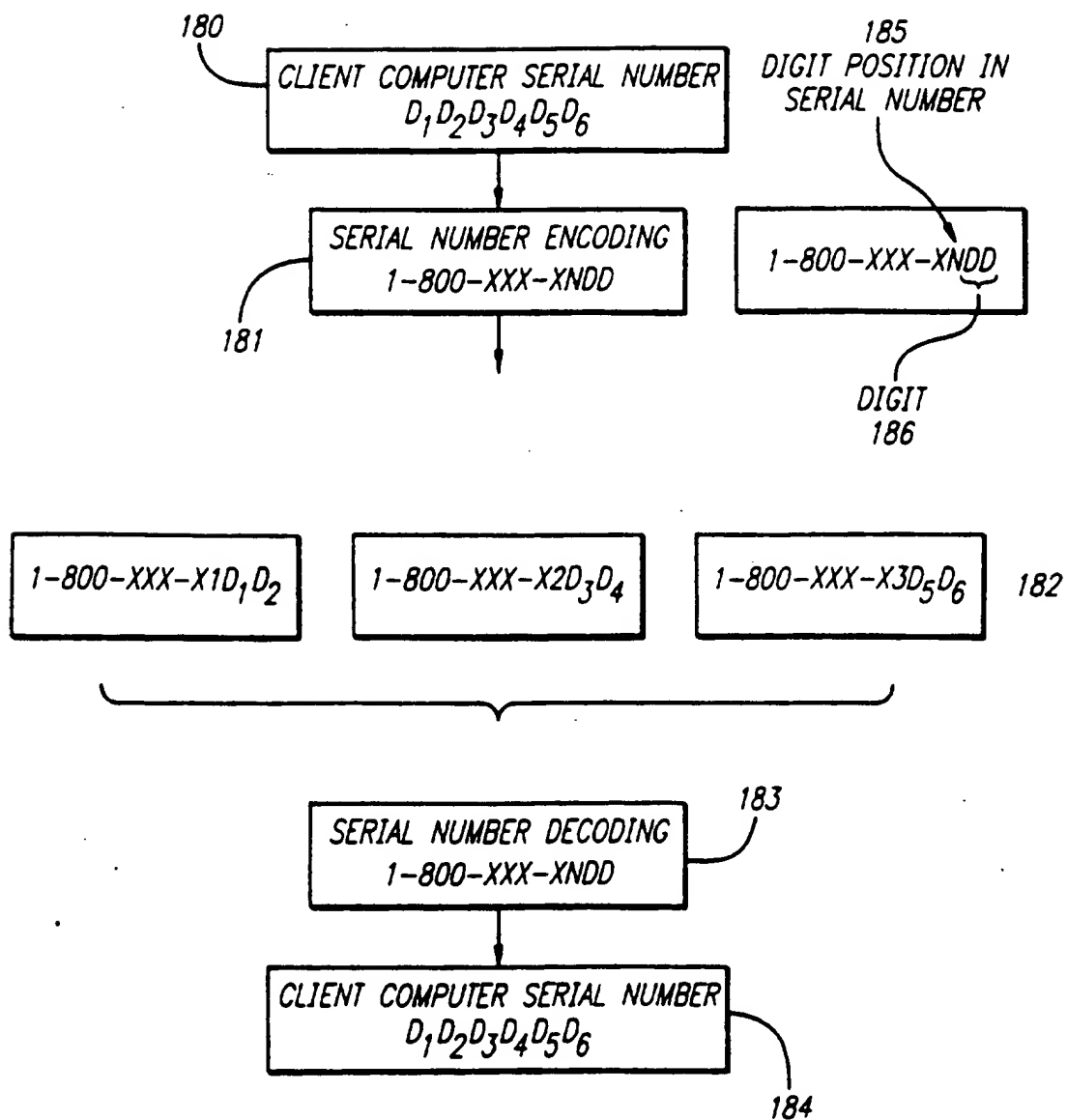


FIG. 4



18/18

FIG. 4A



INTERNATIONAL SEARCH REPORT

International Application No.

PCT/CA 95/00646

A. CLASSIFICATION OF SUBJECT MATTER
IPC 6 G06F1/00 G08B25/01

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Maximum documentation searched (classification system followed by classification symbols)

IPC 6 G06F G08B

Documentation searched other than maximum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	US,A,4 999 621 (LOEB) 12 March 1991 see the whole document ---	1-6
Y	EP,A,0 588 519 (AMERICAN TELEPHONE AND TELEGRAPH COMPANY) 23 March 1994 see column 2, line 42 - column 3, line 40 -----	1-6

☐ Further documents are listed in the continuation of box C.☒ Patent family members are listed in annex.

* Special categories of cited documents:

- *A* document defining the general state of the art which is not considered to be of particular relevance
- *E* earlier document but published on or after the international filing date
- *L* document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
- *O* document referring to an oral disclosure, use, exhibition or other means
- *P* document published prior to the international filing date but later than the priority date claimed

- *T* later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
- *X* document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
- *Y* document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
- *A* document member of the same patent family

Date of the actual completion of the international search

29 March 1996

Date of mailing of the international search report

22. 04. 96

Name and mailing address of the ISA

European Patent Office, P.B. 5818 Patentlaan 2
NL - 2280 HV Rijswijk
Tel. (+ 31-70) 340-2040, Tx. 31 651 epo nl,
Fax (+ 31-70) 340-3016

Authorized officer

Reekmans, M

INTERNATIONAL SEARCH REPORT

Information on patent family members

International Application No

PCT/CA 95/00646

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US-A-4999621	12-03-91	NONE	
EP-A-588519	23-03-94	US-A- 5311596	10-05-94
		CA-A- 2104849	01-03-94
		JP-A- 6204998	22-07-94